z/VM and Linux Operations for z/OS System Programmers

- An introduction to z/VM and Linux to z/OS programmers
- Migrate z/OS to a z/VM guest system
- Install and manage Linux on System z

Lydia Parziale
Roy P Costa
Marian Gasparovic
Omar Badreddin
Rodrigo Ceron Ferreira de Castro
Kenneth P Haas
Guillaume Lasmayous
Raymond van Deurs

Redbooks

ibm.com/redbooks

IBM

International Technical Support Organization

z/VM and Linux Operations for z/OS Systems Programmers

April 2008

SG24-7603-00

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (April 2008)

This edition applies to z/VM Version 5, Release 3, modification 0 (product number 5741-A05).

This document created or updated on July 30, 2008.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
AS/400®
BookManager®
DB2®
developerWorks®
DFSMS™
DFSMS/VM™
DFSMSdss™
DirMaint™
DS8000™
ECKD™
eServer™
HiperSockets™
IBM®
Language Environment®
MVS™
NetView®
OMEGAMON®
OS/390®
Parallel Sysplex®
POWER™
PR/SM™
RACF®
Redbooks®
Redbooks (logo) ®
REXX™
RMF™
S/390®
SAA®
System Storage™
System z™
System z10™
System z9®
SystemPac®
SystemView®
Tivoli®
VSE/ESA™
VTAM®
WebSphere®
Workplace™
z/Architecture®
z/OS®
z/VM®
z/VSE™
z10™
z9™
zSeries®

The following terms are trademarks of other companies:

Carta, and Portable Document Format (PDF) are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

NOW, and the NetApp logo are trademarks or registered trademarks of NetApp, Inc. in the U.S. and other countries.

Novell, SUSE, the Novell logo, and the N logo are registered trademarks of Novell, Inc. in the United States and other countries.

Oracle, JD Edwards, PeopleSoft, Siebel, and TopLink are registered trademarks of Oracle Corporation and/or its affiliates.

Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

This IBM® Redbooks® publication discusses z/VM® and Linux® operations from a z/OS® programmer or system programmer's perspective. While there are other books written about many of these topics, you will find that this book gives enough information about each topic to describe z/VM and Linux on System z™ operations to somebody new to both.

This book is intended for z/OS programmers and system programmers who are transitioning into the z/VM and Linux on System z environments and need a translation guide to assist them.

We base this book on our experiences using System z10™ Enterprise Edition, z/VM version 5.3 RSU 0701 and Novell® Suse Linux Enterprise Server (SLES) 10 on System z.

The team that wrote this book

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Lydia Parziale is a Project Leader for the ITSO team in Poughkeepsie, New York, with domestic and international experience in technology management including software development, project leadership, and strategic planning. Her areas of expertise include e-business development and database management technologies. Lydia is a PMP and an IBM Certified IT Specialist with an MBA in Technology Management and has been employed by IBM for over 20 years in various technology areas.

Roy P Costa is an Advisory Systems Programmer at the IBM International Technical Support Organization, Poughkeepsie Center. He has over 25 years of experience in z/VM systems operation and programming. Roy has worked with Linux on System z for more than 5 years and authored several Redbook and Redpaper publications. He has also provided technical advice and support to numerous IBM Redbook publications for the past 13 years.

Marian Gasparovic is an IT Specialist working for IBM Server and Technology group in IBM Slovakia. He worked as an Administrator for z/OS at Business Partner for six years. He joined IBM in 2004 as storage specialist. Now he works as Field Technical Sales Support for System z in the CEMAAS region as a member of a new workload team. He also works part-time for ITSO in Poughkeepsie.

Kenneth P Haas is a Field Technical Sales Specialist (FTSS) in Atlanta, Georgia with the East Region Technical WebSphere® Software Sales group. He recently graduated from Furman University in Greenville, South Carolina with a dual degrees in Computer Science and Business Management. Joining IBM a year ago he has been a part of IBM's vitality hire program spending a full year training in his specialties which include z/VM and Linux, and a range of Websphere Products on System z.

Rodrigo Ceron Ferreira de Castro works at the IBM Linux Technology Center in Brazil since 2004 leading the development of the IBM Installation Toolkit for Linux on POWER™ project. He has given many speeches on Linux and Operations Research in conferences worldwide and holds four patents in the United States Patent and Trademark Office in related fields. Rodrigo is one of the first IEEE Certified Sofware Development Associates (CSDA) in the world.

Omar Badreddin is a Researcher at the IBM Center for Advanced Studies and Research in Ottawa and a member of Complexity Reduction in Software Engineering research group (CRuiSE). Omar has five years of work experience with IBM as a Software Engineer and has delivered a number of ITSO workshops. His area of expertise includes Virtualization, Modeling, BPM, SOA, and Software Globalization

Guillaume Lasmayous is an IT Specialist at the New Technology Center in Montpellier, France. He has 2.5 years of experience in the mainframe area. He is working as a pre-sales technical specialist, supporting customers running new workloads on System z. His area of expertise includes z/VM, Linux and a range of middlewares supported in this environment.

Raymond van Deurs is a z/OS System Programmer at EDB Business Partner ASA in Norway. He has 26 years of experience in the IBM mainframe area.

Thanks to the following people for their contributions to this project:

Mathieu Dalbin and Sylvain Carta®
IT Specialist in Montpellier, France

Rich Conway, Robert Haimowitz
International Technical Support Organization, Poughkeepsie Center

Special thanks to the authors of the IBM Redbooks publication, *Introduction to the New Mainframe: z/VM Basics*, SG24-7316, published in November, 2007:

Edi Lopes Alves, Klaus Egeler, Clive Jordan, Jason Herne, Eli M Dow, Eravimangalath P Naveen, Manoj S Pattabhiraman, Kyle Smith

Become a published author

Join us for a two- to six-week residency program! Help write a book dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Why z/VM and Linux for System z?

In this chapter we explore the benefits of using z/VM and Linux on System z and how you can expolit these benefits.

Objectives

On completion of this chapter you should be able to:

- See the benefits z/VM can provide to your IT shop
- Understand why companies are investing in z/VM and Linux on System z

1.1 Why z/VM ?

z/VM provides businesses with the ability to take full advantage of their IT resources by optimizing their utilization through virtualization. With over 40 years of experience, z/VM has become a leader in the field of virtualization. This section discusses the benefits of using that experience of z/VM to support your IT infrastructure.

1.1.1 The Virtual Machine Capability of z/VM

z/VM presents a unique approach to computer operating systems. It provides each user with an individual working environment known as a virtual machine. The virtual machine uses virualization to simulate the existence of a real machine by sharing resources of a real machine which include processors, storage, memory, and input/output (I/O) resources.

Operating systems and application programs can run in virtual machines as guests. For example, you can run multiple Linux and z/OS images on the same z/VM system that is also supporting various applications and users. As a result, development, testing, and production environments can share a single physical computer.

The virtual machine capability of z/VM allows you to perform the following tasks:

- Test programs that can cause abnormal termination of real machine operations without impacting the processing of production work running simultaneously.

  The isolation that is provided for a virtual machine enables system-oriented programs and teleprocessing applications, for example, to be tested on the virtual machine while production work is in progress, because this testing cannot cause abnormal termination of the real machine.

- Test a new operating system release.

  A new release of an operating system can be generated and tested at the same time that the existing release is performing production work. This enables the new release to be installed and put into production more quickly. The ability to operate multiple operating systems concurrently under z/VM may enable an installation to continue running programs that operate only under a back-level release (programs that are release-sensitive and uneconomical to convert, for example) concurrently with the most current release.

- Test a new operating system.

  The existing operating system can be used to process production work concurrently with the generation and testing of a new operating system. Experience with the new system can be obtained before it is used on a production basis, without dedicating the real machine to this function.

- Perform operating system maintenance concurrently with production work.

  The installation and testing of program temporary fixes (PTFs) for an operating system can be done at the same time that normal production operations are in progress.

- Provide backup facilities for the primary operating system.

  A generated z/VM system is not model-dependent and can operate on various server models as long as the minimum hardware requirements are present. This enables a smaller server model that has less real storage, fewer channels, fewer direct access devices, and fewer unit record devices than a larger server model to provide backup for the larger model (normally at a reduced level of performance).

- Perform operator training concurrently with production work processing.

  The real machine does not have to be dedicated to training additional or new operators or to providing initial training when a new operating system is installed. Operator errors cannot cause termination of real machine operations.

- Simulate new system configurations before the installation of additional channels and I/O devices.

  The relative load on channels and I/O devices can be determined using the simulated I/O configuration rather than the real I/O configuration. Experience with generating and operating an I/O configuration for multiple guests can be obtained using one real machine.

- Test customer-written system exits.

  Customer-written system exits can be tested without disrupting production work.

- Provide the same hardware configuration for guest operating systems

  The consistence of the same hardware means the use of the same drivers everywhere so there is no need to test many different drivers. This reduces the amount of testing and configuration for different guest systems.

1.1.2 z/VM provides proven system integrity, security, and reliability

z/VM is built on a foundation of system integrity and security, and incorporates many design features for reliability and availability.

- Integrity and security:
  - z/VM supports guest use of the cryptographic facilities provided by supported IBM servers.
  - IBM will correct any integrity exposures introduced by unauthorized programs into the system.
  - Kerberos authentication and Secure Sockets Layer (SSL) support are provided through TCP/IP for z/VM.
  - Integrated access control and authentication services can be augmented with the addition of the RACF® for z/VM feature or other external security managers.
- Availability and reliability:
  - Application recovery: z/VM provides services that permit recovery of incomplete interactions with resource managers.
  - Automated operations: z/VM offers several levels of automated system management support. One example is the Programmable Operator. For a higher degree of automation, IBM SystemView® Host Management Facilities/VM can be added. Both the Programmable Operator and Host Management Facilities/VM can interact with IBM NetView® on z/VM, which in turn can interact with NetView on z/OS.
  - z/VM provides duplexed data with transparent ongoing synchronization between the primary and backup copy, and automatic transparent switching to the backup copy in case of an error in the primary copy.
  - Online configuration changes eliminate many previously required outages.
  - z/VM systems can be connected for improved server and user availability.
  - Fast restart reduces the user impact of any outage

1.2 Why Linux ?

Linux on System z is all about helping to remove complexity from your IT infrastructure: reducing server sprawl, keeping a lid on software licensing fees, and minimizing the need for human intervention in managing and maintaining your servers.

Open Source

IBM is committed to open source as both a license and a development model for several reasons:

- IBM clients and partners have requested open source software (including Linux) support for all IBM platforms, products, and solutions.
- Open source software, with its wide distribution and use, typically becomes an industry standard.
- Innovation within the open source community typically occurs at a higher rate and volume than in closed-source communities.

With this said, IBM is a major contributor to the open source community, with over 250 developers worldwide working full time on several initiatives. One initiative is the Linux Technology Center (LTC),which works to enable the enterprise capabilities of Linux through development and contribution of technology, utilities, tools, and code.

You can learn more about the LTC on the Web at:

http://oss.software.ibm.com/linux

1.3 Why Linux & z/OS together with z/VM ?

This section looks at using z/VM in combination with Linux on System z and z/OS. Since z/VM provides a highly flexible test and production environment it becomes very easy way to deploy a fully functional operating system such as Linux on System z or z/OS. This allows for server simplification through server consolidation.

1.3.1 Server simplification

As companies have grown, their IT infrastructures have grown as well. This has caused most companies to increase their number of stand-alone servers, storage devices and applications. This increase can lead to enormous inefficiency issues and system management headaches.

This has lead companies to employ server simplification using virtualization. In simple terms, virtualization offers a way to help consolidate a large number of individual small machines on one larger server, easing manageability and more efficiently using system resources by allowing them to be prioritized and allocated to the workloads needing them most at any given point in time. Thus, you will reduce the need to over-provision for individual workload spikes.[1]

[1] http://www-03.ibm.com/systems/z/advantages/virtualization/index.html

Server simplification outside of System z

With the ability to run Linux on System z, information technology departments are able to simplify their processes by consolidating their multiple server farms down to a single System z running z/VM and Linux on System z. This leads not only to hardware savings but also to power savings and it simplifies managing your infrastructure which frees up some IT resources to focus on more mission critical tasks.

Using virtualization, the ability to bring up Linux systems takes a matter of minutes rather than days waiting for new hardware and the installation process to complete. Since each of these Linux images are running in their own virtual system they do not affect the other systems around them.

Server simplification inside of System z

Server simplification does not always have to happen outside of your System z. Sometimes the best place is right there inside the mainframe, meaning taking other operating systems, such as z/OS, that are on other LPARs and moving them to a single LPAR running z/VM. There are several benefits:

- Free up LPARs for more mission critical environments
- The ability to rapidly bring up and down test and development environments
- Helps reduce the cost of upgrades because you can free up LPARs and not have to buy another System z.

z/VM gives the ability to allow a user to have a spare system to try out an idea or to isolate a task. Since z/VM is capable of making available almost limitless spare systems in virtual machines, it allows your IT department to effectively utilize the whole power of their System z. One of the advantages of making one system look like many is that you can create an exact replica of your production system on which you can test your new programs, services, and procedures. This is a relatively inexpensive way to have your own test system (or as many test systems as you would like). It is also a safe way to test a new function, because your real production system, its applications and data, are protected from any damage that the new function might cause if you tested it on the host system.

1.3.2 Ten great reasons to run Linux as a guest of z/VM

Running the Linux operating system as a guest of z/VM is a smart choice. Consider the following benefits z/VM offers a Linux guest environment[2]:

- Sharing resources

[2] From: http://www.vm.ibm.com/linux/benefits.html

Resources can be shared among multiple Linux images running on the same VM system. These resources include: CPU cycles, memory, storage devices, and network adapters.

- Server hardware consolidation.

  Running tens or hundreds of Linux instances on a single System z server offers customers savings in space and personnel required to manage real hardware.

- Virtualization

  The virtual machine environment is highly flexible and adaptable. New Linux guests can be added to a VM system quickly and easily without requiring dedicated resources. This is useful for replicating servers in addition to giving users a highly flexible test environment. See 2.1.1, “The Benefits of Virtualization” on page 14 for more benefits.

- System z advantages

  Running Linux on VM means the Linux guest(s) can transparently take advantage of VM support for System z hardware architecture and RAS features.

- z/VM Connectivity

  z/VM provides high-performance communication among virtual machines running Linux and other operating systems on the same processor. The underlying technologies enabling high-speed TCP/IP connections are virtual channel-to-channel (CTC) adapter support and VM IUCV (Inter-User Communication Vehicle). Simplification of the network by using HiperSockets™ may provide savings and reduce cabling, hubs, switches, and routers, as well as help to reduce maintenance effort.

- Minidisk driver

  Linux on System z includes a minidisk device driver that can access all DASD types supported by z/VM.

- Data-in-memory

  Data-in-memory performance boosts are offered by VM exploitation of the z/Architecture®.

- Debugging

  VM offers a rich debug environment that is particularly valuable for diagnosing problems in the Linux kernel and device drivers.

- Control and automation

  VM's long-standing support for scheduling, automation, performance monitoring and reporting, and virtual machine management is available for Linux virtual machines as well!

- Horizontal growth

  An effective way to grow your Linux workload capacity is to add more Linux guests to a VM system.

z/VM supports Integrated Facility for Linux (IFL) processors, which can be an attractively-priced hardware feature for Linux workloads running on System z.

2

Introducing z/VM to z/OS system programmers

This chapter starts with a short discussion on some basics of virtualization. It continues with an overview of z/VM. In fact, throughout this chapter, there is an overview of z/VM concepts and features. We also include lists of references for more details on the each of this chapter's subjects.

As you become a z/VM system programmer or system administrator, you will need to understand the concept of virtualization and how z/VM implements virtualization.

Objectives

On completion of this chapter you should be able to obtain:

- A basic understanding of virtualization and how z/VM takes advantage of virtualization
- Knowledge of two z/VM components: Control Program (CP) and the Conversational Monitor System (CMS)
- Some basic navigation skills of z/VM and knowledge of simple commands

2.1 Virtualization Overview

Virtualization is considered to be the ability for a computer system to share physical resources, such as memory, storage, network adapters and CPUs so that it can act as many different systems.

The most common places to see virtualization today are with servers, networks, and storage. Virtualization can also be applied to non-physical resources such as applications, middleware, and even virtual resources themselves ie. virtualizing a cluster of virtual servers.

2.1.1 The Benefits of Virtualization

The cost of administering IT systems is growing faster than the cost of new hardware for those systems because the complexity of those systems requires growing numbers of people to manage them. A primary concern of management is to contain cost while increasing revenue levels.

Introducing virtualization can be a critical first step in managing computing infrastructures:

- by lowering the cost of existing infrastructure
- by reducing the complexity of adding resources to that infrastructure
- by building heterogeneous infrastructure across multiple data centers, making those centers more responsive to business needs

The benefits of virtualization vary, depending on the objectives and the specific virtualization technologies selected as well as on the existing IT infrastructure. Users realize many of the following benefits to some degree, even when using virtualization for simple server consolidation.

Higher resource utilization

Virtualization enables the dynamic sharing of physical resources and resource pools, resulting in higher resource utilization, especially for variable workloads where the average needs are much less than an entire dedicated resource.

Lower management costs

Virtualization can improve staff productivity by reducing the number of physical resources that must be managed; hiding some of the resource complexity; simplifying common management tasks through automation, better information and centralization; and enabling workload management automation. Virtualization also enables common tools to be used across multiple platforms.

Usage flexibility

Virtualization enables resources to be deployed and reconfigured dynamically to meet changing business needs.

Improved security and guest isolation

Virtualization enables separation and compartmentalization that is not available with simpler sharing mechanisms, and that provides controlled, secure access to data and devices. Each virtual machine can be completely isolated from the host machine and other virtual machines. If one virtual machine crashes, none of the others are affected.

Virtualization prevents data from leaking across virtual machines, and ensures that applications communicate only over configured network connections.

Higher availability

Virtualization enables physical resources to be removed, upgraded, or changed without affecting users.

Increased scalability

Resource partitioning and aggregation enable a virtual resource, depending on the product, to be much smaller or much larger than an individual physical resource, meaning that you can make scale adjustments without changes to the physical resource configuration.

Interoperability and investment protection

Virtual resources can provide compatibility with interfaces and protocols that are unavailable in the underlying physical resources. This is increasingly important for supporting existing systems and ensuring backward compatibility as done by z/VM.

Improved provisioning

Virtualization can enable resource allocation to a finer degree of granularity than individual physical units. Virtualized resources, because of their abstraction from hardware and operating system issues, are often capable of recovering much more rapidly after a crash than a physical resource.

Consolidation

Virtualization enables multiple applications and operating systems to be supported in one physical system, as well as consolidating servers into virtual machines on either a scale-up or scale-out architecture, It also enables systems to treat computing resources as a uniform pool that can be allocated to virtual machines in a controlled manner.

2.1.2 How virtualization works

Virtualization deals with enabling basic systems management of multiple, often heterogeneous systems right “out of the box”.

As Figure 2-1 on page 16 shows, partitioning and virtualization involve a shift in thinking from physical to logical by treating IT resources as logical pools rather than as separate physical entities. This involves consolidating and pooling IT resources, and providing a “single system illusion” for both homogeneous and hetrogeneous servers, storage, distributed systems, and networks.

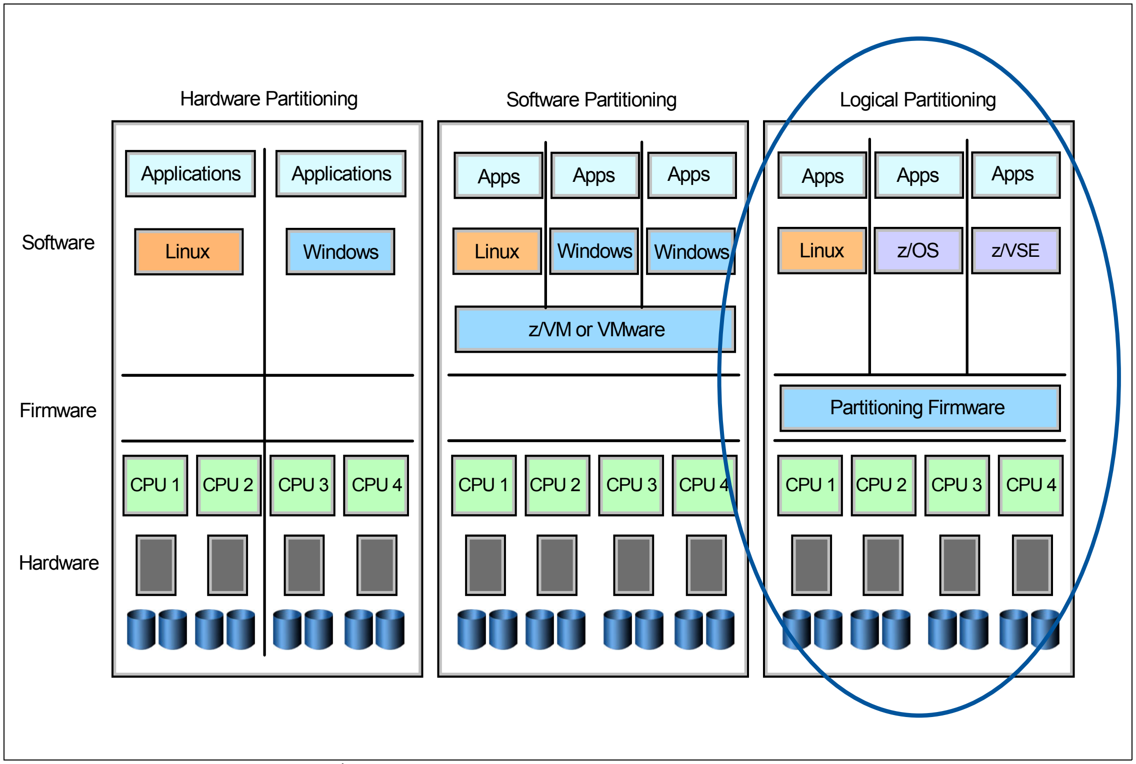

Figure 2-1 Logical partitioning[1]

Partitioning of hardware involves separate CPUs for separate operating systems, each of which runs its specific applications. Software partitioning employs a software-based hypervisor” to enable individual operating systems to run on any or all of the CPUs.

[1] z/VM does not run Windows® as a “guest” operating system because Windows is not a mainframe operating system. Figure 2-1 simply shows the operating systems that run on z/VM or VMWare; it does not show the operating systems which run on both.

Hypervisors are virtualization platforms that allow multiple operating systems to run on a host computer at the same time. Hypervisor technology originated in the IBM VM/370, the predecessor of the z/VM we have today. Logical partitioning (LPAR) involves partitioning firmware (a hardware-based hypervisor) to isolate the operating system from the CPUs.

2.1.3 Virtualization on the Mainframe

Typically, IBM mainframes can be partitioned in three different ways: basic mode, LPAR mode, and z/VM guest.

- In the basic mode of operation also called “native mode”, the entire physical system is used as a single system. The choice of basic mode of operation is selected during the system activation time; however, this is the least used mode of operation.In basic mode, logical partitions (LPAR) are not supported.

> **Note:** On z9™ and z10 mainframes, “basic mode” is no longer supported. It is mandatory to define at least one LPAR.

- In the logically partitioned, or LPAR mode of operation, a single mainframe system is logically divided into multiple partitions. The LPAR mode of operation is the most common mode used on mainframes. Depending on the machine, the LPAR mode may provide additional facilities not available in basic mode, and these facilities can be exploited with operating system support.

> **Note:** The mode of operation is selected during system activation time

- IBM z/VM guest implementations are software level partitioning. The z/VM operating system runs either on an LPAR or, on older hardware, on the complete mainframe (basic mode). Then virtual machines (VM) are created on top of the z/VM system to host other “guest” systems.

2.2 z/VM Overview

Getting used to a new Operating System (OS) usually involves getting to know the different ways of performing the same actions that you are familiar with in another OS. However, sometimes an OS is so different from everything else you have seen that it is worth taking a look at a few basic concepts of the new OS before you actually start exploring the new OS functions and capabilites.

At first glance, z/VM can look very different than other operating systems. However, once you have a grasp on some basic concepts it becomes quite easy to navigate and exploit the powers of this OS. Some of z/VM's powers lie in its ability to provided highly flexible test and production environments for enterprises to deploy their on-demand business solutions. z/VM exploits the IBM z/Architecture and helps enterprises meet their growing demands for multi-user server solutions with a broad range of support for operating platforms such as Linux on System z, z/OS, z/OS.e, VSE/ESA™, z/VSE™, and more. For more information you can refer to the following Web site:

http://www.vm.ibm.com/index.html

In this section of the book, we provide an explanation of what z/VM is and the features and components available in Version 5 Release 3. These topics will be useful to you throughout this book as you begin your experience in z/VM, but a more detailed introduction can be found in the IBM Redbooks publication, *Introduction to the New Mainframe: z/VM Basics*, SG24-7316.

2.2.1 History of z/VM

In the early 1960s a small group of university scientists were working on the idea of a time sharing computer system. The system that was created ran on the IBM processor of that time and provided time sharing but utilized in a more batch like processing environment.

The idea of the virtual machine came about in the mid 1960s with the work on a control program (CP) that could partition real disks into minidisks and handle unit record input/output (I/O) devices for each virtual machine. During that same time, CMS was being developed as a monitor system for file and program manipulation. Rather than integrating the two systems, each were split into their own components. CP would provide a separate computing environment at the machine level for each user. CMS would provide single user services that did not have to worry about allocating or sharing resources.

Since the virtual machine control program needed to simulate the computer systems of that day, each virtual machine would need all the components of a computer. Unique computer identity, a secure location, input and output devices, along with memory and storage would need to be available to each virtual machine. Queuing work (I/O devices) was done by spooling the input or output to the device. Input was prepared on card punch machines where holes were punched into cards that had 50 rows and 81 columns (5081 cards). Then a card reader would sense the location of the punched holes and electronically store the information into the computer's memory. Once the computations were completed, output would be sent to an output device which would have been an impact printers or basic consoles.

We still see the legacy of those devices today in our z/VM systems. Since each virtual machine has it's own equipment and each needs addressability, the same default device addresses and device types are assigned to each machine.

Address 0009 is a 3215 console

Address 000C is a 2540 reader

Address 000D is a 2540 punch

Address 000E is a 1403 printer

Along with the I/O devices, each machine needs central processors and memory to run programs and storage to save data. Processors (virtual CPUs), Memory (virtual storage) and disks (minidisks) complete the components needed for each virtual machine.

As we log on to a z/VM userid (our unique machine identifier) and enter our password (our secure location), we are actually initiating the process of starting a computer system. We first make contact with the Control Program (CP) where we are given the required hardware to run our system. Next, we bring up our system. Be it CMS, Linux, z/OS or another z/VM system, we do an Initial Program LOAD (IPL) of a system that already resides on disk. We may spool input to our punch or reader or spool output to our printer or console.

Over the years, IBM picked up this development effort. Work on VM continued as IBM announced newer processors such as system/360 and system/370, but was not seen as a system that customers would widely utilize. As the base system/370 operating system, MVS™ continued it's development efforts and there were so few processor prototypes available, it was quickly seen that VM provided a needed processor simulation platform for the MVS developers. VM became a popular system among many customers that found the ease of programming and versatility of the system beneficial to their business.

By the 1980s, VM installations grew and more business processes and applications were being developed to run on VM. Customer popularity brought about enhanced programming products for VM such as REXX™ and Pipelines which has since been ported to other operating systems. As new IBM processors and architectures were developed, VM kept pace with new functions to exploit the features.

With the advent of other technologies and systems, z/VM was becoming less of a front end direct user interface and more as IBM's virtualization technology platform. Now with the popularity of Linux and the recognition of z/VM virtualization capabilities, it has again demonstrated it's place in the System z operating system portfolio.

As with early computers, z/VM has evolved from it's early roots and now provides many more functions and services, but the underlying principles of the virtual machine has not significantly changed.

Figure 2-2 on page 20 shows a timeline of how z/VM has evolved to today's version.

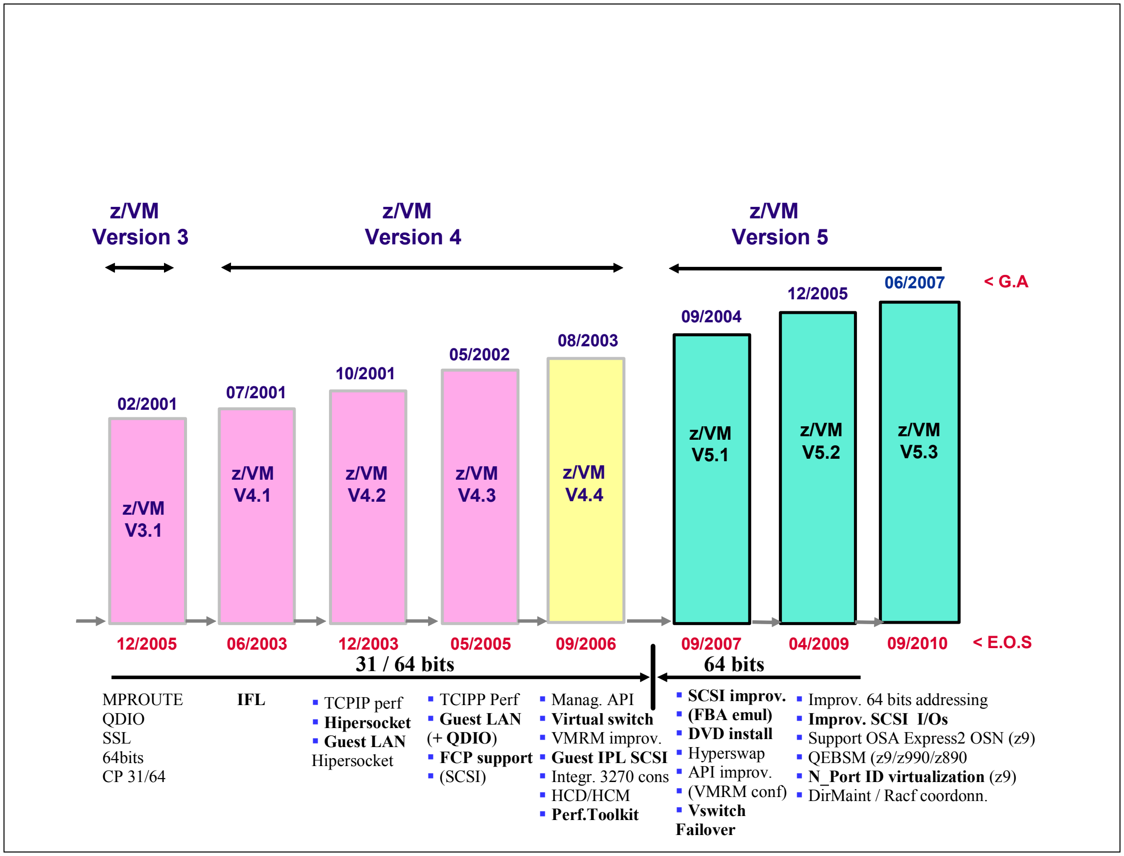

Figure 2-2 History of z/VM

2.2.2 About z/VM

z/VM provides each user with an individual working environment known as a *virtual machine*. The virtual machine simulates the existence of a dedicated real machine, including processor functions, memory, networking, and input/output (I/O) resources. Operating systems and application programs can run in virtual machines on the second level as guest. For example, you can run multiple Linux and z/OS images on the same z/VM system each on their own virtual machine.

As a result, development, testing, and production environments can run side by side sharing the same resources.

A virtual machine can use its own real hardware resources (like a tape drive); however, for the most part it shares from a pool of virtual hardware managed by z/VM, the virtual address of these devices may or may not be the same as the real address of the actual device. Therefore, a virtual machine only knows "virtual hardware" that may or may not exist in the real world.

2.2.3 First-Level versus second-level guest system

A first-level z/VM means that it is the base operating system that is installed directly on top of the real hardware. A second-level system is a user brought up in z/VM where an OS can be executed upon the first-level z/VM.

In other words, there is a first-level z/VM operating system that sits directly on the hardware, but the guests of this first-level z/VM system are virtualized; see Figure 2-3 on page 22. By virtualizing the hardware from the first-level, we are able to create and use as many guests as needed with a small amount of actual real hardware.

Operating systems running in these virtual machines are often called "guests". Other terms and phrases you might encounter include:

- "Running first level" meaning running directly on the hardware (which is what a base z/VM does).
- "Running second level", "running under z/VM", or "running on (top of) z/VM" means running as a guest on the second-level.

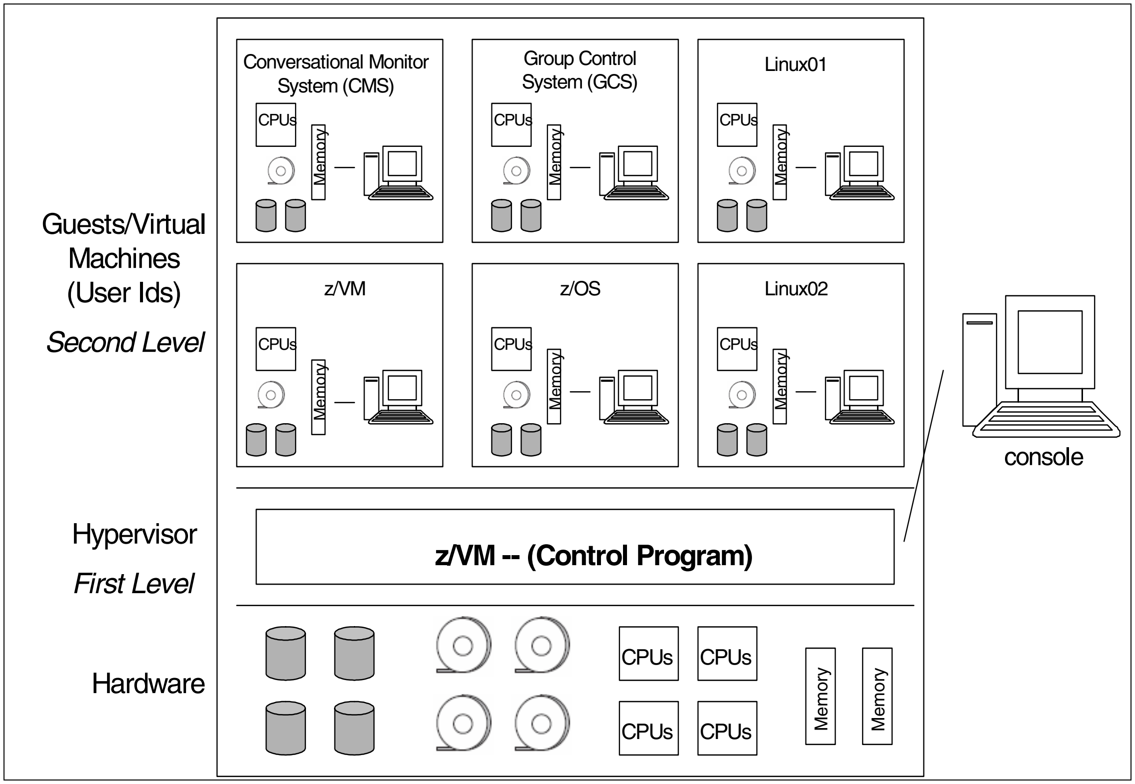

Figure 2-3 Example of configuration of an LPAR

2.2.4 User Directory

The user directory (USER DIRECT) is a file containing all the information about all of the system's guests. It is usually owned by the MAINT user on disk 2CC.

> **z/OS analogy:** The user directory can be seen as the equivalent of all user definitions in the security system, or contents of the data set SYS1.UADS

When a user definition is created, we need to specify the CP privilege classes, Minimum/Maximum Virtual memory that a user can access, and the mode in which the VM is going to run (for example, ESA/XA) as well as the I/O devices it will need.

Example 2-1 on page 23 shows an entry for the user named IBMUSER1 in the user directory.

Example 2-1 USER DIRECT entry for user IBMUSER1

```
USER IBMUSER1 USER1PW 32M 128M G           1
  INCLUDE IBMDFLT                          2
  MACH XA                                3
  IPL CMS                                  4
  MDISK 191 3390 1823 010 LX6RES  MR       5
```

The entries that make up the z/VM guest definition in USER DIRECT have the following meanings:

1. The USER line sets the user ID of this virtual machine, which is *IBMUSER1*. The password is *USER1PW*. This machine is defined as having a default memory storage of 32 MB when it logs in. This user could increase the memory allocation to a maximum of 128MB by using the **DEFINE STORAGE** command. The last value represents the CP classes of a user ID (in our example - *G* for general user authority).

> **Note:** The DEFINE STORAGEcommand is a disruptive and will reset a guest operating system.

2. Use a default profile for common user attributes as shown in Example 2-2 on page 24.
3. Run this virtual machine in XA mode.
4. At logon, initalize Conversational Monitor System (CMS) automatically.
5. Establish a minidisk for the user (in our example - with virtual address 191 on a 3390 DASD volume at cylinder 1823 for 10 cylinders on DASD volume LX6RES with multiwrite access..

If certain directory control statements are repeated for several users, you can make use of directory profiles to save space in the directory. Because you can potentially create many hundreds of Linux z/VM userids on a single z/VM image, you can create a profile to define the things that many user IDs have in common. The profile we included in our TCPIP userid is shown in Example 2-2.

Example 2-2 Include profile IBMDFLT

```
PROFILE IBMDFLT
  SPOOL 000C 2540 READER *    1
  SPOOL 000D 2540 PUNCH A     1
  SPOOL 000E 1403 A           1
  CONSOLE 009 3215 T          1
  LINK MAINT 0190 0190 RR     2
  LINK MAINT 019D 019D RR     2
  LINK MAINT 019E 019E RR     2
  LINK MAINT 0402 0402 RR     2
  LINK MAINT 0401 0401 RR     2
  LINK MAINT 0405 0405 RR     2
```

The following statements, displayed in Example 2-2, are in the IBMDFLT include profile:

1. The SPOOL statements define the user’s virtual spool devices. The options you select modify operational functions associated with your virtual reader, printer, punch, or console.
2. Provide access to files that resides on other users disks, in this case MAINT disks that contain CMS, tools and help files.

2.3 Components of z/VM

z/VM consists of the following components and facilities (base products):

- Control Program (CP)
- Conversational Monitor System (CMS)
- Transmission Control Protocol/Internet Protocol (TCP/IP) for z/VM
- Advanced Program-to-Program Communication/Virtual Machine (APPC/VM) Virtual Telecommunications Access Method (VTAM®) Support (AVS)
- Dump Viewing Facility
- Group Control System (GCS)
- Hardware Configuration Definition (HCD) and Hardware Configuration Manager (HCM) for z/VM
- Language Environment®
- Open Systems Adapter Support Facility (OSA/SF)
- Restructured Extended Executor/Virtual Machine (REXX/VM)

- Transparent Services Access Facility (TSAF)
- Virtual Machine Serviceability Enhancements Staged/Extended (VMSES/E)

In addition to these basic products, z/VM also offers the following optional features:

- Data Facility Storage Management Subsystem for VM (DFSMS/VM™)
- Directory Maintenance Facility for z/VM (DirMaint™)
- Performance Toolkit for VM
- RACF Security Server for z/VM
- Remote Spooling Communications Subsystem (RSCS) Networking for z/VM

In the following sections we discuss each component and point you to references where you can learn more about each topic.

2.3.1 Control Program

Control Program (CP) is primarily a real-machine resource manager. CP provides each user with an individual working environment known as a “virtual machine”. Each virtual machine is a functional equivalent of a real system, sharing the real processor function, storage, console, and input/output (I/O) device resources.

When you first log on to z/VM, CP controls the working environment. Many of the facilities of z/VM are immediately available to you. For example, you can use CP commands to do various system management tasks. However, most of the work done on z/VM requires the Conversational Monitor System (CMS). From this environment, a guest operating system (such as z/OS) can be IPLed and run within the userid.

CP provides connectivity support that allows application programs to exchange information with each other and to access resources residing on the same z/VM system or on different z/VM systems.

CP is discussed further in section 2.4, “Control Program (CP)” on page 37.

z/OS analogy: CP is the equivalent of z/OS NUCLEUS or SUPERVISOR

2.3.2 Conversational Monitor System

Conversational Monitor System (CMS) provides a high-capacity environment that supports large numbers of interactive users. CMS can help you perform a wide variety of tasks. For example, you can write, test, and debug application programs for use on CMS or a guest system's base product, as listed here:

- Run application programs developed on CMS or guest systems
- Create and edit data files
- Process jobs in batch mode
- Share data between CMS and guest systems
- Communicate with other system users
- Provides a useful file system for storing data

CMS File System

The *file* is the essential unit of data in CMS. Files in CMS are unique and cannot be read or written using other operating systems. When you create a file in CMS, you name it using a file identifier (file ID). The file ID consists of three fields:

1. File Name (fn) - maximum of 8 alpha-numeric characters
2. File Type (ft) - maximum of 8 alph-numeric characters
3. File Mode (fm) - 1 alphabetic character

z/OS analogy: A CMS file is like a sequential data set on z/OS or a z/OS file on USS.

When you use CMS commands and programs to modify, update, or refer to files, you must identify the file by using these fields. Some CMS commands allow you to enter only the file name, or the file name and file type; others require you to enter the file mode as well.

Under z/VM, your files can be stored within a Shared File System (SFS) (a shared file space) which is in a hierarchical directory structure. Depending on your system configuration, you may have the option to use both CMS and SFS methods for storing files. In this situation, you could store those files that you may want to share in your SFS file space; other files could be stored on minidisks.

z/VM OpenExtensions (POSIX support) includes another type of file called a byte file system (BFS) file. BFS files are organized in a hierarchy, as in a UNIX® system. All files are members of a *directory*. Each directory is in turn a member of another directory at a higher level in the hierarchy. The highest level of the hierarchy is the BFS file space. Typically, a user has all or part of a BFS file space mounted as the root directory.

z/VM views an entire file hierarchy as a byte file system. Each byte file system is a mountable file system. The *root* file system is the first file system mounted. Subsequent file systems can be mounted on any directory within the root file system, or on a directory within any mounted file system.

All files in the byte file system are called BFS files. BFS files are byte-oriented, rather than record-oriented, like CMS record files on minidisks or in the Shared File System. You can copy BFS files into CMS record files, and you can copy CMS record files into the Byte File System.

CMS is discussed further in section 2.5, “Conversational Monitor System (CMS)” on page 39.

2.3.3 TCP/IP

TCP/IP for z/VM brings the power and resources of your mainframe server to the Internet. Using the TCP/IP protocol suite of TCP/IP for z/VM, you can reach open, multivendor networking environments from your z/VM system.

TCP/IP for z/VM allows z/VM systems to act as peers of other central computers in TCP/IP open networks. Applications can be shared transparently across z/VM, Linux, and other environments. Users can send messages, transfer files, share printers, and access remote resources across a broad range of systems from multiple vendors

TCP/IP for z/VM provides the following types of functions:

- Connectivity and gateway functions, which handle the physical interfaces and routing of data
- Server functions, which provide a service to a client (that is, send or transfer a file)
- Client functions, which request a certain service from a server anywhere in the network
- Network status and management functions, which detect and solve network problems
- Application programming interfaces, which allow you to write your own client/server application

z/OS analogy: TCP/IP for z/VM has the same functionality as in z/OS.

2.3.4 APPC/VM VTAM Support (AVS)

AVS (APPC/VM VTAM Support) is a Virtual Telecommunications Access Method (VTAM) application that provides advanced program-to-program communication (APPC) services between VM and non-VM systems in an SNA network.

AVS and VTAM run in the same GCS group on a z/VM system. Together, AVS and VTAM enable APPC/VM application programs in a TSAF or communication services (CS) collection to communicate with the following applications:

- Other APPC/VM applications residing in other VM systems within the SNA network
- APPC applications residing in non-VM systems in the SNA network

z/OS analogy: Same functionality in z/OS

2.3.5 Dump Viewing Facility

The Dump Viewing Facility (DVF) helps you interactively to diagnose system problems. Using this facility, you can display, format, and print data interactively from virtual machine dumps, as well as display and format recorded trace data.

The BLOCKDEF utility lets you display, format, and print control block information. The VIEWSYM command lets you display symptom records, making it easier to identify duplicate problems when they occur.

z/OS analogy: DVF is similar to the Interactive Problem Control System (IPCS) in z/OS

2.3.6 Group Control System (GCS)

Group Control System (GCS) runs in an XA or XC virtual machine in place of CMS. It is a virtual machine supervisor, providing multitasking services that allow numerous tasks to remain active in the virtual machine at one time.

One of the functions of GCS is to support a native Systems Network Architecture (SNA) network. The SNA network relies on ACF/VTAM, VTAM SNA Console Support (VSCS), and other network applications to manage its collection of links between terminals, controllers, and processors. GCS provides services for ACF/VTAM, VSCS, and the others, which eliminates your need for VTAM Communications Network Application (VM/VCNA) and a second operating system like VSE.

2.3.7 HCD and HCM for z/VM

HCD and HCM for z/VM, or Hardware Configuration Definition and Hardware Configuration Manager for z/VM, provide a comprehensive I/O configuration management environment, similar to that available with the z/OS operating system.

HCM runs on a Windows-based personal computer connected to the z/VM system through a TCP/IP network connection. HCM provides a graphical user interface, as well as commands, to help you configure your system. You supply the needed I/O configuration information to HCM, which processes the information and passes it to HCD.

HCD runs in a z/VM server virtual machine and performs the work of actually creating and changing the hardware and software aspects of your I/O configuration.

Although HCM provides the primary user interface to HCD, HCD also provides a backup user interface on your z/VM host for certain I/O configuration tasks, in case HCM is not available.

The original dynamic I/O configuration capabilities of z/VM are still valid. These consist of a set of system operator commands for changing the hardware server's I/O configuration while the system continues to run, or for managing the hardware I/O configuration of all of the logical partitions in your server.

You now have the choice of either using these commands, or using HCM and HCD, to manage your I/O configuration. Note, however, that the use of HCM and HCD is incompatible with the original dynamic I/O configuration capabilities. You should select one method to use for the duration of any given IPL of your z/VM system.

z/OS analogy: HCD and HCM for z/VM have the same functionality as in z/OS.

2.3.8 Language Environment

Language Environment (LE) provides the runtime environment for programs written in C/C++, COBOL, or PL/I. Language Environment helps you create mixed-language applications and gives you a consistent method of accessing common, frequently-used services.

Language Environment consists of:

- Basic routines that support starting and stopping programs, allocating storage, communicating with programs written in different languages, and indicating and handling conditions.
- Common library services, such as math services and date and time services, that are commonly needed by programs running on the system. These functions are supported through a library of callable services.

- Language-specific portions of the runtime library. Because many language-specific routines call Language Environment services, behavior is consistent across languages.

z/OS analogy: LE has the same functionality as in z/OS.

2.3.9 OSA/SF

The Open Systems Adapter-Express (OSA-Express) and Open Systems Adapter Express2 (OSA-Express2) are integrated hardware features that allow the System z platform to provide industry-standard connectivity directly to clients on local area networks (LANs) and wide area networks (WANs).

The Open Systems Adapter Support Facility (OSA/SF) is a host-based tool supplied with z/VM that allows you to customize an OSA's modes of operation. You can access OSA/SF by a CMS user ID, by a REXX call to the OSA/SF API, or through a Java™-based graphical user interface (GUI).

z/OS analogy: OSA/SF has the same functionality in z/OS.

2.3.10 REXX/VM

REXX/VM contains the REXX/VM Interpreter, which processes the English-like Restructured Extended Executor (REXX) programming language. It also contains the z/VM implementation of the SAA® REXX programming language. REXX/VM provides a single source base for the REXX/VM Interpreter in the CMS and GCS components. The REXX/VM Interpreter exploits 31-bit addressing.

The REXX/VM Interpreter helps improve the productivity of your organization. Using REXX, you can write customized application programs and command procedures, tailor CMS commands, and create new XEDIT macros.

z/OS analogy: There are some minor differences between REXX/VM and REXX in z/OS. For instance: The default REXX environment, when invoking a REXX EXEC in z/VM is CMS, in z/OS it would be TSO.

2.3.11 TSAF

TSAF, or Transparent Services Access Facility, provides communication services within a collection of VM systems without using VTAM. TSAF runs in a CMS virtual machine.

A group of up to eight VM systems that each have TSAF installed and running can form a TSAF collection. APPC/VM programs on one VM system in the TSAF collection can communicate with other APPC/VM programs on the other VM systems in the collection. The routing is transparent to the application programs. Communications between the applications proceed as if the applications were running on the same system.

2.3.12 VMSES/E

VMSES/E, or Virtual Machine Serviceability Enhancements Staged/Extended, helps you install z/VM and other VMSES/E-enabled products and apply code changes that correct or circumvent reported problems. VMSES/E handles both source code and object code.

VMSES/E can also help you define, build, and manage saved segments. The VMFSGMAP command provides a saved segment mapping interface that lets you modify saved segment definitions and view saved segment layouts prior to actually building them on your system.

The next section provides overviews of the optional features of z/VM.

2.3.13 DFSMS/VM

DFSMS/VM, or Data Facility Storage Management Subsystem for VM, allows you to control your data and storage resources more efficiently. DFSMS/VM provides the following support.

z/OS analogy: DFSMS/VM has similar functionality to z/OS DFSMS™

Space management

DFSMS/VM improves DASD utilization by automatically managing space in SFS file pools. As the SFS administrator, DFSMS/VM allows you to:

- Convert SFS storage to DFSMS-managed storage by assigning management classes to files and directories. Each management class tells DFSMS/VM how to treat its members in the course of its management of the file pool.

- Automatically manage files based on the criteria in each management class. This management may consist of deletion of files, automatic migration of files, or both.
- Migrate (or move) files from DFSMS-managed storage to DFSMS-owned storage by using the assigned management class. This function also compresses the data. The files can be automatically recalled when referenced (opened and browsed), or they can be explicitly recalled.

Minidisk management

Using DFSMS/VM for minidisk management allows you to check the integrity of CMS minidisks and move them from one location to another. DFSMS/VM helps you migrate CMS minidisks to new DASD quickly, efficiently, and with minimal impact to users.

Interactive Storage Management Facility (ISMF)

DFSMS/VM uses the ISMF to provide a consistent user interface for storage management tasks.

IBM Tape Library Dataserver Support

DFSMS/VM provides native VM support for the IBM 3494 and 3495 Tape Library Dataservers.

2.3.14 Directory Maintenance Facility for z/VM

The Directory Maintenance Facility for z/VM (DirMaint) provides efficient and secure interactive facilities for maintaining your z/VM system directory. Directory management is simplified by DirMaint's command interface and automated facilities. DirMaint provides a corresponding command for every z/VM directory statement, including Cross System Extensions (CSE) cluster directory statements. DirMaint's error checking ensures that only valid changes are made to the directory, and that only authorized personnel are able to make the requested changes.

DirMaint offers the following functionality:

- DirMaint operates as a CMS application and uses CMS interfaces for CMS and CP services. As a CMS application, DirMaint is not dependent on specific hardware, although it does verify that the device types specified in DirMaint commands are only those supported by the z/VM host.
- DirMaint functions are accomplished by two disconnected virtual machines equipped with an automatic restart facility. The use of virtual machines takes

advantage of the inherent reliability, availability, and serviceability of the system architecture.

- Any transaction requiring the allocation or deallocation of minidisk extents can be handled automatically.
- All user-initiated transactions can be password-controlled and can be recorded for auditing purposes.
- Command authorization is controlled by assigning DirMaint commands to privileged command sets. Users may be authorized to issue commands from multiple command sets. DirMaint provides nine predefined command sets, but up to 36 sets are supported.
- User exit routines enable centralized directory maintenance of remote systems. Some exit routines also enable DirMaint to interact with other facilities, such as RACF.
- The open command structure allows you to replace all commands with your own user-written commands.
- An automated process for copying CMS minidisk files minimizes the possibility of human error. This process optionally formats the old (source) minidisk before returning it to the available minidisk pool.
- The integrity of CMS files is ensured by preventing new minidisk space from being inadvertently allocated over existing extents.
- DirMaint improves overall system efficiency by minimizing the number of DIRECTXA utility runs required. The update-in-place facility (DIAGNOSE code X'84') can be used to place many of the changes online immediately.
- System security is enhanced by providing the ability to enforce regular password changes. When changing the password, the user is required to enter the new password twice to guard against typographical errors.
- An additional level of security can be implemented by requiring that a password be entered for every user transaction. This is the default.

2.3.15 Performance Toolkit for VM

The Performance Toolkit for VM™, which is derived from the FCON/ESA program (5788-LGA), assists operators and systems programmers or analysts in the following areas:

- Operation of the system operator console in full screen mode
- Support for managing multiple VM systems
- Post-processing of VM history files
- Performance monitoring
- Serving data through a Web server for viewing with Web browsers

- Personal computer-based graphics
- TCP/IP performance reporting

In addition to analyzing VM performance data, the Performance Toolkit processes Linux performance data obtained from the Resource Management Facility (RMF™) Linux modular data gatherer, **rmfpms**. The gathered data can be analyzed using the RMF PM client application. The Linux performance data obtained from RMF can be viewed and printed in a manner similar to the presentation of VM data.

The **rmfpms** application is available from the following site:

http://www.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/pmweb/pmlin.html

> **z/OS analogy:** Similar to RMF MONITOR in z/OS

2.3.16 RACF Security Server for z/VM

The RACF Security Server for z/VM, or Resource Access Control Facility, is a security tool that works together with existing functions in the z/VM base system to provide improved data security for an installation. RACF protects information by controlling access to it.

RACF also controls what you can do on the operating system and protects your resources. It provides this security by identifying and verifying users, authorizing users to access protected resources, and recording and reporting access attempts.

To help each installation meet its unique security needs and objectives, RACF provides:

- Flexible control of access to protected resources
- The ability to store information for other products
- A choice of centralized or decentralized control profiles
- Transparency to end users
- Exits for installation-written routines

Your organization can define individuals and groups who use the system that RACF protects. A security administrator uses RACF to define a profile for each individual that identifies that person's user ID, password, and other information.

A *group* is a collection of individuals who have common needs and requirements. For example, a whole department may be defined as one group. Your organization can also define what authorities you have, or what authorities a group you belong to has. RACF controls what you can do on the system. Some individuals have a great degree of authority, while others have little authority. The degree of authority you are given is based on what you need to do your job.

In addition to defining user and group authorities, RACF protects resources. You can protect system resources and user resources. System resources include system minidisks, system SFS files and directories, certain VM events, and terminals. User resources include user minidisks and user SFS files and directories.

RACF stores all this information about users, groups, and resources in profiles. A *profile* is a record of RACF information that has been defined by the security administrator. There are user, group, and resource profiles.

Using the information in its profiles, RACF authorizes access to certain resources. RACF applies user attributes, group authorities, and resource authorities to control use of the system. The security administrator or someone in authority in your organization controls the information in your user profile, in group profiles, and in resource profiles. You, as an end user, control the information in profiles describing your own resources, such as your own minidisks. You can protect your data by setting up resource profiles. You can set up an access list in your resource profile to control who has read-access and who has write-access to your data.

In addition to uniquely identifying and authorizing users, RACF can record what users do on the system. It keeps track of what happens on the system so that an organization can monitor who is logged on to the system at any given time. RACF reports if persons have attempted to perform unauthorized actions. For example, RACF can record when someone who does not have the proper authority tries to use or change your data. The security administrator can monitor these activities and generate reports.

z/OS analogy: RACF for z/VM has the same functionality as RACF for z/OS.

2.3.17 RSCS Networking for z/VM

RSCS Networking for z/VM, or Remote Spooling Communication Subsystem, commonly referred to as RSCS, is a networking program that enables users on a z/VM system to send messages, files, commands, and jobs to other users within a network. RSCS connects nodes (systems, devices, and workstations) using links. These links allow data to be transferred between the nodes.

Running under the GCS component of z/VM, RSCS uses the spooling facilities of z/VM to store and retrieve data. z/VM handles data transfer within its system by means of spooling. RSCS extends the basic spooling capabilities of z/VM, handling data transfer between the z/VM system and outside sources. Data is stored on a spool after RSCS receives it and until RSCS can forward it to its destination. RSCS uses communications equipment to transfer data between the local z/VM system and other systems or remote locations

A node in an RSCS network is either a system node or a station node. A station node can originate and receive information. It can be a computer, a workstation, or a printer. A system node, however, must be a computer. In addition to originating and receiving information, system nodes can also relay information between two other nodes.

RSCS can communicate with system nodes that are running under the control of network job entry (NJE)-compatible subsystems, such as:

- JES2 or JES3
- RSCS
- VSE/POWER
- AS/400® Communications Utilities
- Products that provide NJE functions for Linux or AIX®

RSCS can communicate with the following types of station nodes:

- ASCII printers or plotters
- Computers running under the control of a system that can provide a multileaving protocol
- IBM 3270 Information Display System Printers
- Line printer router (LPR) daemons and clients in a TCP/IP network
- Unsolicited File Transfer (UFT) daemons and clients in a TCP/IP network
- Workstations running under the control of remote job entry (RJE)

Each link in an RSCS network is associated with a programming routine, called a *driver*, that manages the transmission and reception of files, messages, and commands over the link. The way that a driver manages the data is called a *protocol*. All file transmission between networking nodes uses NJE protocol, 3270 printers use 3270 data streams, workstations use RJE protocol, and ASCII printers use data streams appropriate to that printer.

Systems Network Architecture (SNA) provides one set of protocols that governs communications on links. The method that RSCS uses for sending data to a node varies, depending on the type of connection used to establish the link. RSCS can support non-SNA (such as binary synchronous communication or channel-to-channel), SNA, and TCP/IP connections.

z/OS analogy: RSCS networking is similar to JES2 networking in z/OS

2.4 Control Program (CP)

z/VM is comprised of two primary components: Control Program (CP) and Conversational Monitor System (CMS). The rest of z/VM's compents are discuss later on in this chapter. In this section, we explain the concept and function of CP.

CP is the operating system that underlies all of z/VM. CP is responsible for virtualizing your System z machine's real hardware and allowing all the virtual machines to simultaneously share that real hardware resource.

CP also handles the creation and management of virtual machines. It can be considered the operating system component of z/VM because it is responsible for managing real devices and resources and sharing them among various tasks and users that need them.

z/OS analogy: CP is analogous to the NUCLEUS or SUPERVISOR in z/OS or the kernel in a Linux operating system.

As a resource manager, CP can provide you with a set of virtual machines which can run any operating system that would not ordinarily run on your System z hardware. Without CP you could only be running one operating system on your hardware at any given time (logical partitioning aside). CP allows you to have multiple virtual machines (also known as “guests”), and each one can be running an instance of an operating system simultaneously. As shown in Figure 2-4 on page 38, CP allows you to run many Linux and z/OS images simultaneously on the same piece of hardware.

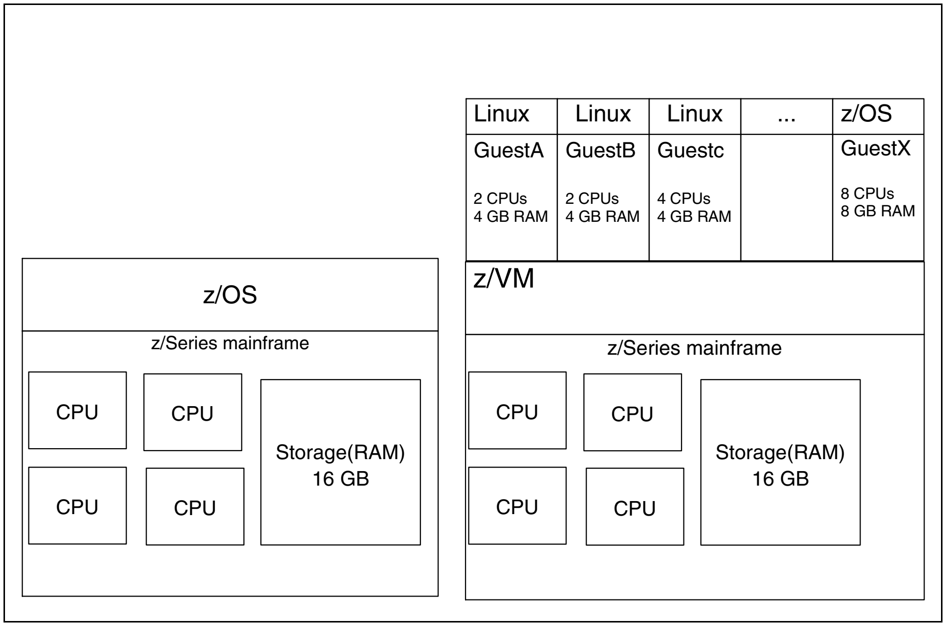

Figure 2-4 CP executing many guest operating systems

Notice in the figure that, even though we only have 4 CPUs and 16 GB of Real Storage, our guest virtual machines are collectively defined to have 16 CPUs and 20 GB of Storage. This is because the CPUs and Storage in a guest's operating system are virtual. You can create a guest that has 64 CPUs if you prefer, and CP will take turns executing tasks on behalf of all 64 virtual CPUs on all of the available real processors. The CP components that handle this task are known as the *scheduler* and the *dispatcher*.

z/OS analogy: These components are very much like the process scheduler and dispatcher of z/OS or the scheduler of Microsoft® Windows and GNU/Linux, except they schedule entire virtual machines for execution instead of simply processes.

Note that the ability to overcommit resources like this only works because an operating system typically does not need 100% of the resources allocated to it all the time that it is running. Sometimes that operating system will be idle, and when it is, another operating system can use the hardware resources.

Virtual machines are defined in a text file known as the user directoy as discussed in 2.2.4, "User Directory" on page 22. For each virtual machine on the system, there is a section in the directory that describes that virtual machine. One of these descriptions is known as a directory entry.

The directory entry for a guest details its guest name, its privileges to execute CP commands, the number of CPUs it has, the amount of memory it has, as well as information on which virtual devices are defined for this particular VM. It is the job of the system administrator to set up and maintain the user directory, and general users typically do not have access to it.

2.4.1 CP modes of execution

At any time, your virtual machine can be in one of two basic states of execution.

- Running a guest operating system (guest mode, or emulation mode)
- Not running a guest operating system (CP mode)

When a guest operating system is running, it is controlling the devices and resources of your virtual machine. Any commands you issue to the virtual machine through your terminal command line will likely be interpreted by the guest operating system and not CP.

However, when your virtual machine is *not* running a guest operating system, CP has control and your processors (as well as your other virtual devices) are generally sitting idle and not consuming real machine resources. In this state, any commands issued to the virtual machine through your terminal command line will be interpreted by CP and not a guest operating system.

Before you start a guest operating system, you are in CP mode. When you start a guest operating system, you are in guest mode. If you shut down your guest operating system, you will be returned to CP mode. Your virtual machine can only be in one of these modes at any instance in time.

You are, however, free to switch between these modes at will, even when your guest operating system is running. If you switch to CP mode while a guest operating system is running, that operating system will be frozen and will not run again until you leave CP mode and go back to guest mode. At that point the guest operating system will continue running as if it had never been interrupted.

2.5 Conversational Monitor System (CMS)

The Conversational Monitor System (CMS) is the most commonly used point of interaction with z/VM. CMS expresses the vast power of the mainframe through a rich set of commands and utilities that build on the toolset CP provides. It is an operating system designed to facilitate mainframe virtual machine administration, by providing users an environment with a higher level of functionality than CP.

CMS executes on top of CP just as any other another operating system that can run on CP.

z/OS analogy: CMS is similar to TSO in z/OS and user space in LINUX

CMS can help you perform a wide variety of tasks such as writing, testing, and debugging application programs for use on CMS or guest systems; executing application programs developed on CMS or guest systems; creating and editing data files; sharing data between CMS and guest operating systems; and communicating with other system users.

Note: CMS is a powerful operating environment, but be aware that CMS *cannot* run at the same time as another guest operating system—only CP can do that.

2.5.1 Using the help command in CMS

This section explains the use of the **HELP** command in CMS. It covers task, component, and command menus, and discusses formatting options. It mentions other ways to invoke help, and describes how to handle messages. Finally, this section explains how to exit the help system in CMS.

The first CMS command we cover here is probably the most useful: **HELP**. This CMS command provides information about many standard system tasks. It also provides the syntax and function of all CMS and CP commands, as well as the meaning of error messages that may be reported to users.

Use the **HELP** command if you cannot remember the format of a command, or if you need information about what a command does. The output will display a substantial amount of information that is also contained in official IBM documentation.

The **HELP** command is similar to the **man** command found on Linux systems, and is simple to use. If you type **help** by itself, you are offered a menu of further options, along with a brief description of each option. By default, typing **help** on the command line will bring you to the task-based help menu.

The next few sections of this chapter explain how to access the **HELP** command more directly. To select an entry from the screen, place your cursor under the appropriate word and press Enter. From there you are taken to another help screen, such as a task menu or a command menu.

> **z/OS analogy:** The **HELP** command is similar to the TSO HELP or to the HELP panels in ISPF

Task menus

Task menus are the portion of the HELP system that provide an index and description of tasks that you may want to perform, including creating, modifying or changing files, or customizing the CMS installation. Task menus provide you a way to obtain details about specific actions, without requiring you to know the name of the command in advance.

When you reach the deepest level of navigation relating to a particular topic of interest, the bottom “branch” provides the description and proper format of the required command.

To view the list of tasks and components available to you, type:

```
help task
```

Figure 2-5 on page 41 shows the output of a **help task** command.

```
 HELP TASK                  Task Help Information                 line 1 of 39
 (c) Copyright IBM Corporation 1990, 2004

 z/VM Help, main panel

 This panel lists other Help panels that provide information about
 various z/VM functions, topics, and tasks.
 To view a Help panel, move the cursor to any character of the name
 and press the ENTER key or the PF1 key.


 HELPINFO - HELP Facility topics
 MENUS    - z/VM Help menus
 TASKS    - Basic z/VM tasks - good choice for beginners
 COMMANDS - z/VM commands available to general users
 CMS      - CMS commands
 CP       - CP commands
 QUERYSET - QUERY and SET commands and subcommands
 TCPIP    - TCP/IP commands
 PF1= Help     2= Top      3= Quit     4= Return     5= Clocate    6= ?
 PF7= Backward 8= Forward  9= PFkeys  10=           11=           12= Cursor

====> _
                                                          Macro-read 1 File
MA    a                                                            23/007
```

Figure 2-5 Output of help task command

Component menus

Component menus list the names of all the command HELP files available for a specific HELP component.

To display all the command HELP files available for CMS, start at the HELP TASK menu shown in Figure 2-6. Position your cursor anywhere under the word `CMS` and press Enter.

You will reach the Menu Help Information screen shown in Figure 2-6 on page 42, which displays all the command HELP files available for CMS.

```
 CMS MENU                  Menu Help Information                  line 1 of 42
 (c) Copyright IBM Corporation 1990, 2004

 Help for CMS commands

 To view a Help panel, move the cursor to any character of the name
 and press the ENTER key or the PF1 key.
 An asterisk (*) preceding the name indicates a MENU panel.
 A colon (:) preceding the name indicates a TASK panel.


 *BORDER   CMDCALL  DSERV    GLOBALV  NETLCNVT RT       SYNMSGS
 *CMSQUERY CMSBATCH Edit     GRAnt    NOTE     RTNDrop  SYNonym
 *CMSSET   COMpare  ERASE    HB       NUCXDROP RTNLoad  SYSWATCH
 *CMSUTIL  CONWAIT  ESERV    Help     NUCXLOAD RTNMap   TAPE
 *EDIT     COPYfile ESTATE   HELPCONV NUCXMAP  RTNState TAPEMAC
 *FILESERV CP       ESTATEW  HI       OPNMSGS  RUN      TAPPDS
 *OPENVM   CREate   ETRACE   HO       OPtion   SADT     TE
 *OSHELL   CSLGEN   EXec     HT       OSRUN    SAMGEN   TELL
 PF1= Help     2= Top      3= Quit      4= Return     5= Clocate   6= ?
 PF7= Backward 8= Forward  9= PFkeys   10=           11=          12= Cursor

====> _
                                                      Macro-read 2 Files
MA    a                                                            23/007
```

Figure 2-6 CMS component menu

Command menus

In addition to the task menus provided in the help system, there are separate command menus for a number of VM components, including one large menu that covers CP and CMS.

If you select the command menu for CMS from the initial help menu, for example, a panel showing a list of available CMS commands appears. Menu navigation is similar to the task menus; to choose a command, place the cursor over it and press Enter or PF1.

Formatting options

The HELP Facility provides various ways of getting information for a command, depending on your level of expertise and the amount of detail you require for a particular task. Commands can contain three layers of information: BRIEF, DETAIL, and RELATED. Each layer displays a unique level of HELP.

You can display any of the three available layers by specifying the corresponding layering options: BRIEF, DETAIL, or RELATED. You may specify only one layering option at a time. However, after you have requested one layer of HELP on a specified command, you may toggle (switch) between the other layers available for that command.

BRIEF is the default option, meaning that if you do not specify a layering option, the BRIEF layer of HELP is displayed if it exists. If **BRIEF HELP** is not available for a certain command, **DETAIL HELP** is displayed. The following sections provide more information on the three layers of command HELP.

BRIEF

BRIEF is the first layer of HELP. It is available for many commands. **BRIEF HELP** displays a short description of the requested command, its basic syntax (the command without options), an example, and if applicable, a message telling you that either more or related information is available.

If you are in full-screen CMS and request BRIEF HELP, your screen shows the **HELP** command you entered and just below it, displays the BRIEF HELP information in a window that is displayed on your screen. If you are not in full-screen CMS, your entire screen displays the BRIEF HELP information.

The following example shows how your screen would look if you requested BRIEF HELP for the **SENDFILE** command and are not in full-screen CMS.

```
help cms sendfile (brief
```

The output of this command is shown in Figure 2-7 on page 44.

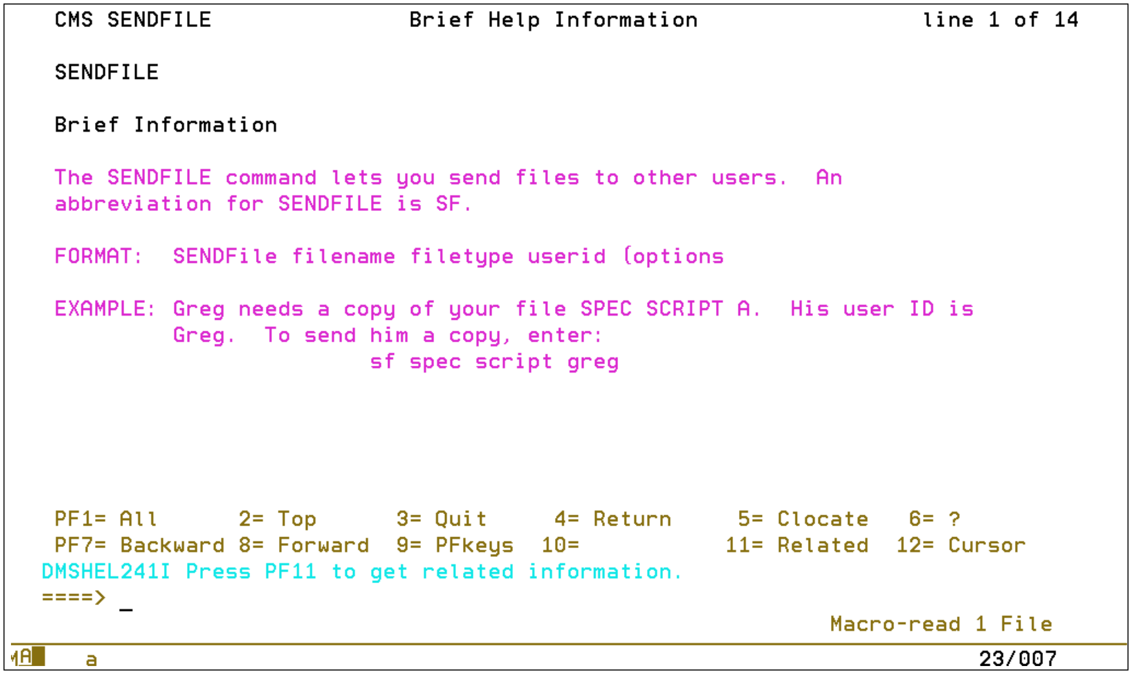

Figure 2-7 Output of the command help cms sendfile (brief

DETAIL

DETAIL provides a complete description of the command, the command format, an explanation of its parameters and options, usage notes, and error information. For more information regarding DETAIL help, refer to *z/VM: CMS Commands and Utilities Reference*.

This layer of HELP has seven subsetting options: **DESCRIPT**, **FORMAT**, **PARMS**, **OPTIONS**, **NOTES**, **ERRORS**, and **ALL**. By specifying subsetting options, you can display one or more particular sections of the detail help.

ALL is the default option, meaning that the entire detail help is displayed. It is possible to change the default option, but if you do so, you will need to specify **ALL** as the subsetting option to display the entire detail layer. For more information about the subsetting options and the **DEFAULTS** command, refer to *z/VM: CMS Commands and Utilities Reference.*

For example, to display the entire DETAIL layer of the **SENDFILE** command in the CMS environment, type in the following command:

```
help cms sendfile (detail
```

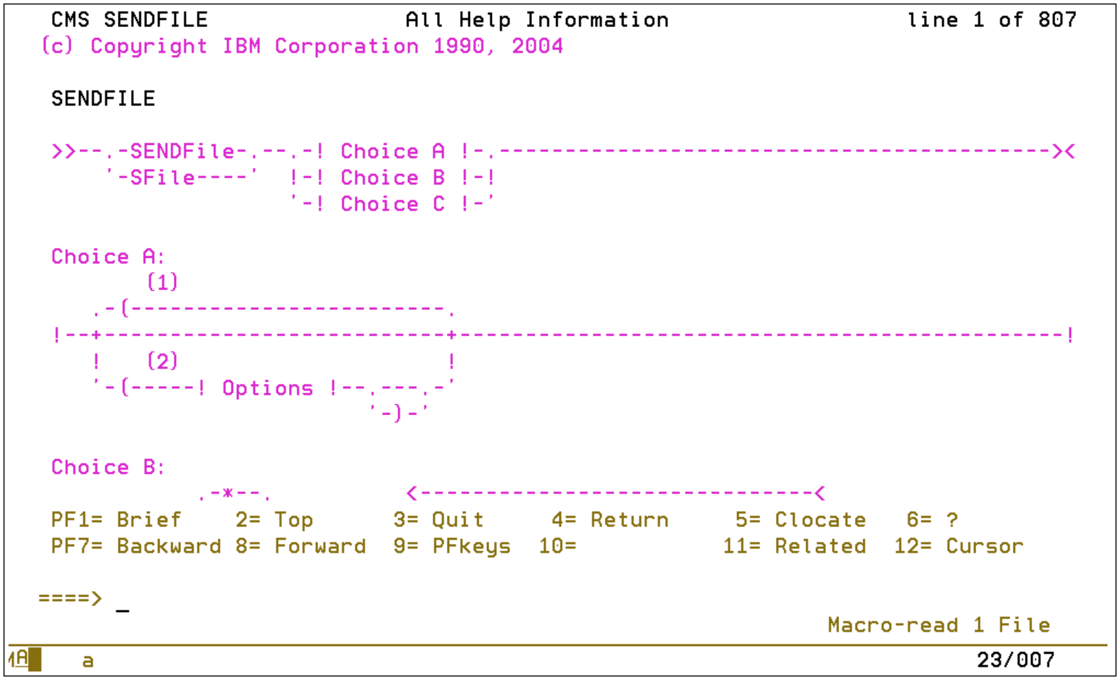

Figure 2-8 Output of the command help cms sendfile (detail

RELATED

The help information entries for some commands allow you to select help entries for similar commands; this is known as RELATED help.

For example, suppose you want to remove a file from your rdrlist. After reading HELP ERASE, you realize that **ERASE** is not the correct command. Instead, using the RELATED layer of the **ERASE** command will let you easily access the HELP file for the correct command, **DISCARD**.

When you request RELATED HELP on the **SET** or **QUERY** commands, the screen lists and briefly describes all the **SET** and **QUERY** operands available for the system component. You can directly access HELP information on any of the displayed operands from these menu screens by positioning the cursor on a particular operand and pressing Enter.

To display the RELATED layer of the **ERASE** command in the CMS environment, enter the following command; Figure 2-9 displays the output of the command.

```
help cms erase (related
```

```
 CMS ERASE                Related Help Information                 line 1 of 20
  Related Information

  For RELATED information on removing files or parts of files from
  your virtual machine, place the cursor under the topic of your choice
  and press ENTER or the PF1 key.


  DELETE     - Removes one or more lines from
               a file while using XEDIT.

  ERASE      - Removes files from your minidisk
               or SFS directory.

  PURGE      - Removes spool files from your
               reader, printer, or punch.

  DISCARD    - Removes files from "list-type"
               CMS command environments, such
 PF1= Help     2= Top      3= Quit     4= Return     5= Clocate    6= ?
 PF7= Backward 8= Forward  9= PFkeys  10= Morehelp  11= Brief     12= Cursor

====> _
                                                         Macro-read 1 File
MA     a                                                             23/007
```

Figure 2-9 Output of help cms erase (related

Other help options

There are five other options that affect the display of **HELP**: **SCREEN**/**NOSCREEN**, **TYPE**/**NOTYPE**, and **EXTEND**. Briefly, these other options control the display of files and error messages and the search order of commands. For complete descriptions of these options, refer to *z/VM: CMS Commands and Utilities Reference, SC24-6073*.

Other ways to get help

You can also get help for a specific command directly, without going through the various forms of help navigation discussed so far. If you know the command you want to see help for, or want to navigate directly to it, specify the command type as shown here:

Help CP MENU - Displays the full screen menu of available CP commands

Help CP command - Displays the format of the specified CP command

Dealing with error messages

Sometimes, when you perform a z/VM task, the system responds with a message. You can use HELP for messages to find out why the message was produced, and perform any necessary corrective action.

The HELP files for messages display the message text, an explanation of why the message was displayed, the system action, and a user action. For example, suppose you receive the following error message:

```
DMSERD107S  Disk 'A'(191) is full
```

You wonder whether this message is significant, so you want more detailed information. For this particular error indicator, issue the help command, followed by the initial identifier present in the message `DMSERD107S`, as shown:

HELP DMSERD107S

or

HELP DMS107S

> **Note:** The module identifier (characters 4 to 6 of the message identifier) is ignored by **HELP**, so you do not need to enter it.
>
> For example, to display information about message DMSHLP002E, you can enter any of these commands:
>
> **help msg dmshlp002e**
> **help msg dms002e**
> **help dmshlp002e**
> **help dms002e**

If you receive a message without a message ID, it could be because you have issued the **CP SET EMSG TEXT** command to display only message text (or an application program might have issued the command).

To obtain information about a message with no message ID, you will need to use help facilities beyond the scope of those built in to CMS (such as the PDF or BookManager® version of the appropriate messages book).

Tips when using HELP

Some commands may not work as expected in HELP mode. For this reason, we recommend that you do *not* enter commands on the command portion of the HELP screen where they are displayed.

In general, use the HELP Menu only for help, and not as a command prompt.

Exiting the HELP system

To exit the help system in CMS at any time, use PF3 or type `quit` on the command line. Note that you may need to issue the **`quit`** command multiple times if you have descended into the HELP Menu hierarchy and want to back up one level at a time.

As an alternative, you can use PF4 to completely leave the HELP system with one keystroke.

> **z/OS analogy:** Navigating the HELP menu is similar to navigating the z/OS HELP menu.

2.6 Networking options in z/VM

The System z brand offers many solutions to improve your network connectivity. Some of the key aspects about networking are the features implemented on System z and z/VM which are OSA adapters, HiperSocket and VSWITCH.

2.6.1 OSA adapters

The Open Systems Adapter (OSA) is a network controller that you can install in a mainframe I/O cage. The adapter integrates several hardware features and supports many networking transport protocols, including fast Ethernet, gigabit Ethernet, 10 gigabit Ethernet, Asynchronous Transfer Mode (ATM), and token ring.

There are several versions of the OSA available: OSA-2, OSA-Express, and OSA-Express2, each of which have slightly different features and functionality. The older OSA-2 cards have been superseded by the newer OSA-Express and OSA-Express2 cards and are no longer available, but many older mainframes still use them.

Network connectivity for the z/VM system or any of its guest operating systems could be provided by directly dedicating specific OSA addresses to the system. This can be done by the DEDICATE statement in the user directory or by **`ATTACH`**ing the devices to the guest userids.

2.6.2 HiperSockets

HiperSockets technology is available for System z servers. HiperSockets is an integrated any-to-any virtual TCP/IP network that provides inter connectivity among multiple LPARs or z/VM guests. With HiperSockets, you can connect any server running on the same System z, and improve response time due to low latency.

HiperSockets was developed to provide highly available, high-speed network connections through a memory bus. It can be considered for an application or infrastructure scenarios, where the network is stressed by a large quantity of data exchanged between servers running on the same System z server. It also provides higher security because packets do not leave the box so they cannot be sniffed on the LAN.

Figure 2-10 on page 50 shows a possible implementation of HiperSockets over an applications server scenario. When you use a Linux for System z technology to produce an application environment, you usually plan to have a three-layer architecture composed by:

- Front layer: HTTP server
- Middle layer: Application server
- Back-end layer: Database server

With the proposed architecture, the environment can be planned with:

- IBM HTTP Server (powered by Apache) under Linux for System z
- IBM WebSphere Application Server under Linux for System z
- DB2® server under z/OS

If a session is started by a browser over the network, a communication between the three layers is started. To avoid network traffic, implementing a HiperSockets solution can help you. With benefits provided by HiperSockets, your network traffic between the layers is exchanged in the System z server at memory speeds.

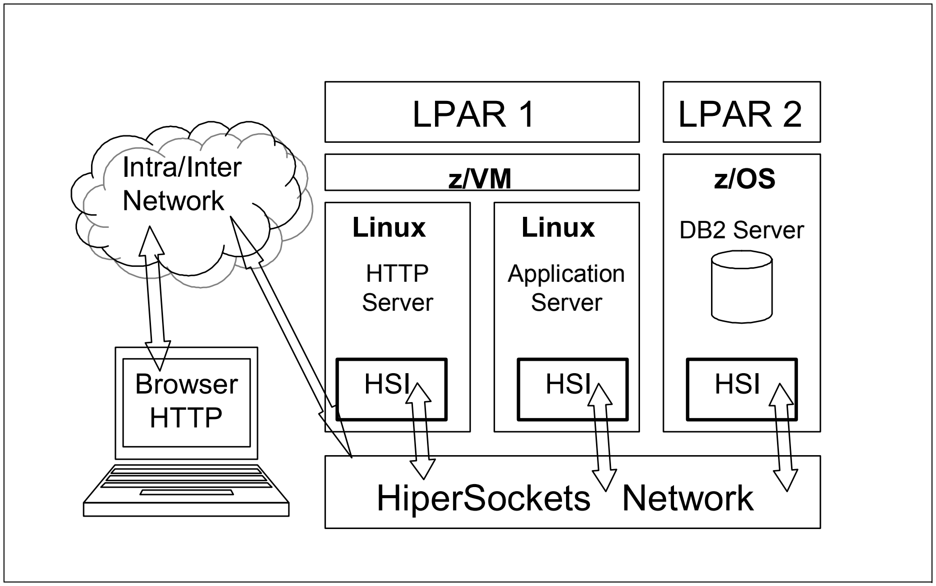

Figure 2-10 The application server scenario with HiperSockets

More on Hipersockets is available in section 6.3, “Networking access” on page 301.

2.6.3 HiperSockets for fast LPAR communication

HiperSockets provides the fastest TCP/IP communication between consolidated Linux, z/VM, z/VSE, and z/OS virtual servers on a System z server. HiperSockets provides internal “virtual” Local Area Networks, which act like TCP/IP networks within the System z server. This integrated Licensed Internal Code (LIC) function, coupled with supporting operating system device drivers, establishes a higher level of network availability, security, simplicity, performance, and cost effectiveness than is available when connecting single servers or LPs together using an external TCP/IP network.

The HiperSockets function, also known as internal Queued Direct Input/Output (iQDIO) or internal QDIO, is an integrated function on the System z servers that provides users with attachment to high-speed “logical” LANs with minimal system and network overhead.

HiperSockets eliminates the need to utilize I/O subsystem operations and the need to traverse an external network connection to communicate between LPs in the same System z server. HiperSockets offers significant value in server consolidation connecting many virtual servers, and can be used instead of certain coupling link configurations in a Parallel Sysplex®.

HiperSockets is customizable to accommodate varying traffic sizes. Since HiperSockets does not use an external network, it can free up system and network resources, eliminating attachment costs while improving availability and performance.

HiperSockets Connectivity

HiperSockets are accessible among combinations of logical partitions (LPs) or virtual servers within the System z. This "network within the box" concept minimizes network latency and maximizes bandwidth capabilities between z/VM, Linux on System z, z/VSE, and z/OS images, or combinations of these. It is also possible to have HiperSockets under z/VM, which permits internal networks between guest operating systems, such as many Linux servers, for example.

Each HiperSockets LAN uses a Channel Path ID (CHPID) with a channel type IQD. Figure 2-11 on page 52 shows an example of HiperSockets connectivity using four HiperSockets with CHPIDs FC, FD, FE, and FF.

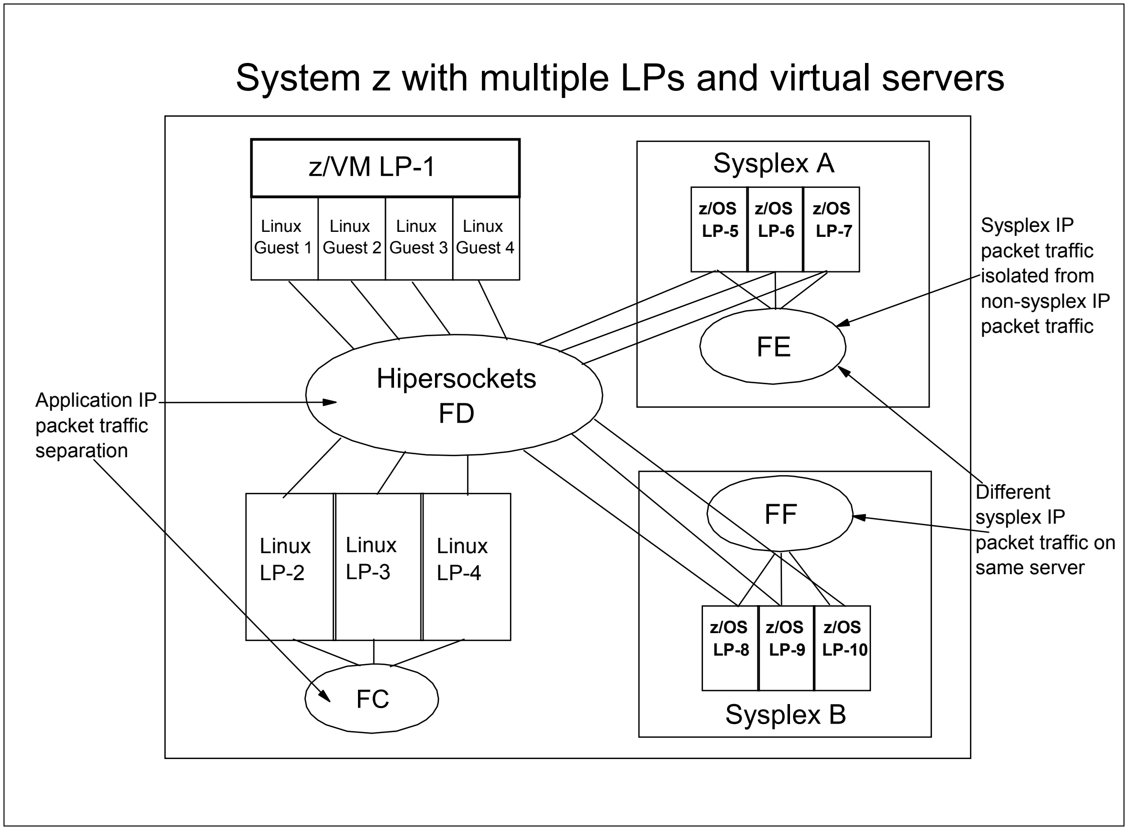

Figure 2-11 HiperSockets connectivity

HiperSockets benefits

System z HiperSockets is a technology that provides high-speed TCP/IP connectivity between virtual servers running within different logical partitions (LPs) of a System z server. It eliminates the need for any physical cabling or external networking connections between these virtual servers.

The virtual servers form a “virtual LAN.” Using iQDIO, the communication between virtual servers is through I/O queues set up in the system memory of the System z server. Traffic between the virtual servers is passed at memory speeds. HiperSockets supports multiple virtual LANs, which operate as TCP/IP networks within a System z server. Section 6.6.1, “HiperSockets Connectivity” on page 291 discusses the number of HiperSockets available for each System z server.

HiperSockets is a LIC function of the System z server. It is supported by the following operating systems:

- All in-service z/OS releases
- All in-service z/OS.e releases
- All in-service z/VM releases
- All in service z/VSE releases
- Linux on System z

There are a number of benefits gained when exploiting the HiperSockets function:

- HiperSockets can be used to communicate among consolidated servers in a single System z server platform. All the consolidated hardware servers can be eliminated, along with the cost, complexity, and maintenance of the networking components that interconnect them.
- Consolidated servers that have to access corporate data residing on the System z server can do so at memory speeds, bypassing all the network overhead and delays.
- HiperSockets can be customized to accommodate varying traffic sizes. (In contrast, LANs like Ethernet and Token Ring have a maximum frame size predefined by their architecture.) With HiperSockets, a maximum frame size can be defined according to the traffic characteristics transported for each of the possible HiperSockets virtual LANs.
- Since there is no server-to-server traffic outside the System z server, a much higher level of network availability, security, simplicity, performance, and cost effectiveness is achieved as compared with servers communicating across an external LAN. For example:
  - Since HiperSockets has no external components, it provides a very secure connection. For security purposes, servers can be connected to different HiperSockets. All security features, like firewall filtering, are available for HiperSockets interfaces as they are for other TCP/IP network interfaces.
  - HiperSockets looks like any other TCP/IP interface; therefore, it is transparent to applications and supported operating systems.
- HiperSockets can also improve TCP/IP communications within a sysplex environment when the DYNAMICXCF facility is used.

For an in-depth discussion on Hipersockets, refer to the *IBM System z Connectivity Handbook*, SG24-5444.

2.6.4 VSWITCH

Virtual switch, or VSWITCH, is the newest LAN type supported by z/VM. It was introduced back in release 4.4 of z/VM and is a special type of guest LAN that can provide external LAN connectivity through Hipersockets or an OSA-Express device without the need for a routing virtual machine.

z/VM does require that one particular machine own the OSA-Express devices that are used by the VSWITCH. These special virtual machines are called VSWITCH controllers, and are essentially extra TCP/IP stacks that manage the OSA on behalf of the systems connected to the VSWITCH. In older releases of z/VM, there was a single predefined VSWITCH controller. Starting with z/VM 5.1 and newer versions there are two predefined VSWITCH controllers: DTCVSW1 and DTCVSW2.

Another benefit of a VSWITCH is that it can easily be configured with redundant OSA devices and additional controllers so that in the event of a problem, either the switch can fail over to a backup OSA or controller.

Figure 2-12 shows a logical view of a virtual switch and where it fits.

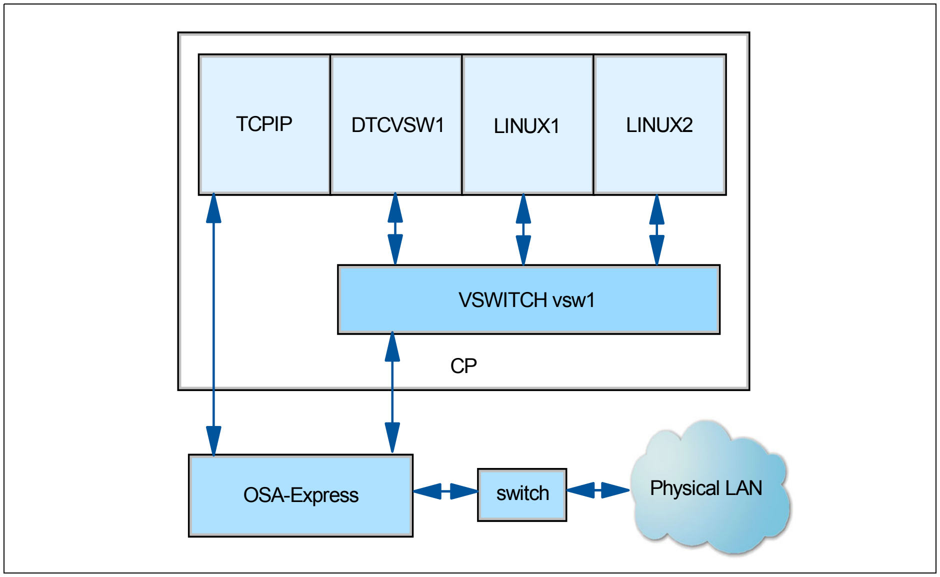

Figure 2-12 Architecture digram for a virtual switch

2.6.5 Further Information on Networking

For futher information on networking refer to the following references:

- *Linux on IBM eServer zSeries and S/390: Large Scale Linux Deployment*, SG24-6824
- *Introduction to the New Mainframe: z/VM Basics*, SG24-7316
- *Networking Overview for Linux on zSeries*, REDP-3901

2.7 Consoles

As a z/VM operator, you may use several types of consoles. Consoles can be located on the same physical device, such as a PC workstation using 3270 emulator software or on physically connected 3270 terminals. Because of the flexibility of z/VM, the number of consoles varies by customer installation. Depending on your installation, there might be one production system or many production systems. There can be hundreds of virtual machines, each running a production system. Regardless of the number of consoles, the following are the four types of consoles available and a description of each follows:

- Hardware Management Console
- Primary system operator console
- Virtual Console for each virtual machine
- Production system's system console when there is a production system running in a virtual machine.

Although the number of consoles at each site depends on the installation, each installation has at least one Hardware Management Console, at least one primary system operator console, and one or more virtual consoles to control virtual machines. A production system often has its own system console.

z/OS analogy: The Hardware Management Console and the primary system operator console are similar to the HMC which is used to access a z/OS LPARs.

For more information on z/VM consoles and how to interact with each of them, please see *z/VM System Operation, SC24-6121* PDF[2].

2.7.1 The Hardware Management Console

A Hardware Management Console (HMC) communicates with each central processor complex (CPC). There can be one or more Hardware Management Consoles depending on your site's installation. This section describes only the specific tasks for the z/VM operator. For additional information on using the Hardware Management Console, see the processor complex hardware publications. The *System z Hardware Management Console Operations Guide* (SC28-6857-01) can be found at:

http://www-1.ibm.com/support/docview.wss?uid=isg28bab02cf07906582852571 1800547c1a&aid=1

[2] http://publibz.boulder.ibm.com/epubs/pdf/hcsf8b22.pdf

From the Hardware Management Console, you can:

- IPL (load) z/VM
- Monitor messages and send commands to z/VM (Operating System Messages panel)
- Perform basic z/VM system management functions
- Install, service, and operate z/VM using the integrated 3270 console

z/OS analogy: z/OS users should be already familiar with this type of console since it is also used to define LPARs, a common task between both z/OS and z/VM, and to access a z/OS console.

Using the Operating System Messages Panel

From the Hardware Management Console, you can use the Operating System Messages task (panel), which is the `SYSC` device, to log on to the *OPERATOR* userid and view z/VM messages. You can also use the Operating System Messages panel issue commands on the z/VM. This communication between the Hardware Management Console and z/VM will be in line mode.

To use the Operating System Messages panel as the system operator console when IPLing the z/VM system, do one of the following:

- Specify the **`CONS=SYSC`** IPL parameter when using the stand-alone program loader (SAPL). See: Passing IPL Parameters in z/VM: CP Planning and Administration.
- Specify a load parameter of **`CONSSYSC`** to bypass the SAPL screen and have the z/VM operator come up on the Operator System Messages Panel. See: Overriding Stand-Alone Program Loader Defaults in z/VM: CP Planning and Administration[3].
- Add **`SYSTEM_CONSOLE`** to the *OPERATOR_CONSOLES* statement in the SYSTEM CONFIG file. See: System Configuration File in z/VM: CP Planning and Administration.

To have emergency operating system message displayed, do the following:

- Add **`SYSTEM_CONSOLE`** to the *EMERGENCY_MESSAGE_CONSOLES* statement in the SYSTEM CONFIG file.

Setting up and using the Integrated 3270 Console

From the Hardware Management Console you can use the integrated 3270 console as the system operator's console. The integrated 3270 console is identified by z/VM as *SYSG*. This console provides you the ability to install,

[3] http://www.vm.ibm.com/pubs/hcsg0b01.pdf

service, and operate z/VM without any attached 3270 devices. To identify the integrated 3270 console to CP when IPLing the z/VM system, do one of the following:

- Use the **CONS=SYSG** IPL parameter when using the stand-alone program loader (SAPL).
- Specify a load parameter of **CONSSYSG** to bypass the SAPL screen and have the z/VM operator come up on the Inegrated 3270 Console. See: *Overriding Stand-Alone Program Loader Defaults in z/VM: CP Planning and Administration*.
- Add **SYSTEM_3270** to the OPERATOR_CONSOLES system configuration statement.

2.7.2 The Primary System Operator Console

The primary system operator console is the console where CP automatically logs on the system *OPERATOR* virtual machine during IPL. This can be the Hardware Management Console, or a device specified during the IPL. If SYSC is specified, the Operating System Messages panel becomes the system operator console. If SYSG is specified, the integrated 3270 console becomes the system operator console. Your installation chooses what device or what Hardware Management Console function (SYSC or SYSG) is identified as the primary system operator console using one of the following:

- SYSTEM CONFIG statement, **OPERATOR_CONSOLES**
- IPL parameter, **CONS=***xxxx*
- Load parameter **CONS**xxxx

The primary system operator console is where you perform most of the tasks in this book. For example, you enter CP commands from the primary system console to perform the following tasks:

- Bringing up the z/VM CP
- Shutting down the z/VM CP
- Controlling z/VM devices
- Communicating with z/VM users
- Responding to z/VM errors
- Collecting information about z/VM operation.

After CP logs on the system **OPERATOR**, it controls the format of the primary system console's screen. If you are unfamiliar with this format, see the *z/VM Virtual Machine Operation, SC24-6128*[4] book.

[4] Found inside the PDF http://publibz.boulder.ibm.com/epubs/pdf/hcsf8b22.pdf

2.7.3 The z/VM Virtual Consoles

Other consoles you use are the virtual consoles for the virtual machines from which you create and initialize work or production systems. You use this console when:

- You log on a production system's virtual machine
- You set up any special running environment it requires
- You use the CP IPL command to load production systems in the virtual machine
- You need to recover a production system.

You use this console to control a production system's virtual machine. To control the production system itself, use the production system's system console

When you log on a production system's virtual machine at this console, CP controls the format of the console's screen. If you are unfamiliar with this format, see the *z/VM: Virtual Machine Operation, SC24-6128* book.

> **Note:** For z/OS guests, when the system console is responding to z/VM errors or IPLing the system, the virtual machine console can be used as an IPL and error-recording console. For more information see the *z/VM: Virtual Machine Operation, SC24-6128* book and the *z/VM: Running Guest Operating Systems, SC24-6115*[a] book.

a. See http://publibz.boulder.ibm.com/epubs/pdf/hcsf8b22.pdf

For a more details plesase refer to "Getting to Know your Virtual Console" on page 59.

2.7.4 The Virtual Machine Guest Systems Console

When there is another operating system running work in a virtual machine, this is referred to as a guest system. You use the guest system's system console to control the work, just as if you were running the guest system directly on the hardware. Some guest systems have their system console on the same physical device as the virtual machine operator's console; other installations might use separate devices.

2.7.5 Getting to Know your Virtual Console

To log on to the z/VM system, type in your user ID and password in the corresponding areas on the z/VM logon screen (shown in Figure 2-13 on page 59), and then press **Enter**.

You can also log on by positioning the cursor in the command line, typing `LOGON` followed by your user ID, then pressing Enter as shown in Figure 2-13. For further detail on how to logon refer to "Logging on" on page 63.

```
z/VM ONLINE



                            / VV          VVV MM       MM
                           /  VV        VVV  MMM     MMM
               ZZZZZZ     /   VV      VVV   MMMM   MMMM
                  ZZ     /    VV    VVV    MM MM MM MM
                 ZZ     /     VV  VVV     MM  MMM  MM
                ZZ     /      VVVVV      MM   M   MM
               ZZ     /       VVV       MM       MM
              ZZZZZZ /        V        MM       MM

                  built on IBM Virtualization Technology



Fill in your USERID and PASSWORD and press ENTER
(Your password will not appear when you type it)
USERID   ===>
PASSWORD ===>

COMMAND  ===>
                                                           RUNNING   VMLINUX6
```

Figure 2-13 Logging on using the command line

The system prompts you for the password (see Example 2-3).

Example 2-3 Prompt for the password

```
LOGON MAINT
ENTER PASSWORD  (IT WILL NOT APPEAR WHEN TYPED):
```

After you are logged in, you may be presented with the CP READ prompt. Press **`Enter`** again and the CP will load CMS.

CMS then executes the *PROFILE EXEC* file of the z/VM user ID, if it exists. In the *PROFILE EXEC*, you can define settings for your z/VM user ID, such as define PF keys, link to MDISKs or IPL a guest operating system like Linux or z/OS.

```
LOGON TCPMAINT
z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0024 RDR,   NO PRT,   NO PUN
LOGON AT 14:50:27 EDT WEDNESDAY 06/13/07
z/VM V5.3.0    2007-05-02 16:25

DMSACC724I 191 replaces A (191)
Ready; T=0.01/0.01 14:50:28









_
                                                       RUNNING   VMLINUX6
MA    a                                                            23/001
```

Figure 2-14 z/VM screen after logon

At this point, you are logged on to your z/VM virtual machine. Before discussing some basic commands that can be used to check your virtual machine, we cover some introductory information about VM commands.

Here we use the term “command” generically; it refers to both CP commands and CP utilities. z/VM uses command languages to correspond to the two environments it creates: Control Program and virtual machine.

Use the Control Program (CP) command language when:

- You are a z/VM system operator and you want to control the resources of the real machine located in your computer room.
- You are a virtual machine user and you want to control your virtual machine’s configuration and environment.

Use a virtual machine command language when:

- You are communicating with the operating system you loaded into your virtual machine.
  - To perform production or test work, load your virtual machine with one of the operating systems supported by the z/VM system. Your virtual machine command language is the command language of the operating system you load. This command language is described in the library that documents that particular operating system.
  - To perform service, installation, and maintenance tasks, along with editing and text creation, communicating with others, and problem solving, load your virtual machine with the conversational monitor system (CMS). CMS is a single user, conversational operating system.

You can use CP commands in the following situations:

- Your virtual machine is in the control program (CP) command environment.

  Your virtual machine is in the CP environment when you log on to z/VM and CP READ is displayed in the lower right corner of the screen.

  On a line-mode ASCII device, there is no status area to display CP READ, so CP is displayed in the output area.
- You press the break key while in full-screen mode before entering a command.

  The break key may be PA1, the VM default break key, or another key that you have defined as the break key using the TERMINAL BRKKEY command. Also, the break key may be totally disabled by some application programs or when in the protected application environment.
- You are in a virtual machine command environment, not running in full-screen mode, and enter the #CP command (and # is your logical line end character).
- You are in the CMS virtual machine environment and enter the CP command.
- You are in the CMS virtual machine environment and have the IMPCP function set ON.

To determine the current command environment on a 3270 device, look at the status area in the lower right corner of the display screen. `CP READ` indicates the CP environment, `VM READ` indicates the virtual machine environment.

If you are running CMS in your virtual machine, `VM READ` indicates the CMS environment. When `RUNNING` appears in the status area, enter a null input line to determine your environment. To enter a null line, press Enter but do not enter any data. When you enter the null line, the status area displays either `CP READ` or `VM READ`.

Also, if you are in a read state, in either the CP command environment (with RUN set OFF) or the CMS command environment, and you enter a null line, the system responds with the name of your command environment: CP or CMS in the system output area.

You enter CP commands using any combination of upper case and lower case letters. When you have typed the command and its operands, press Enter to process the command.

Working in a 3270 terminal

Previously, we explained how to log on to the z/VM system, and you saw the response from the system. Now we discuss what you need to know when working with a 3270 terminal on a z/VM system.

Figure 2-15 displays the layout of the 3270 screen.

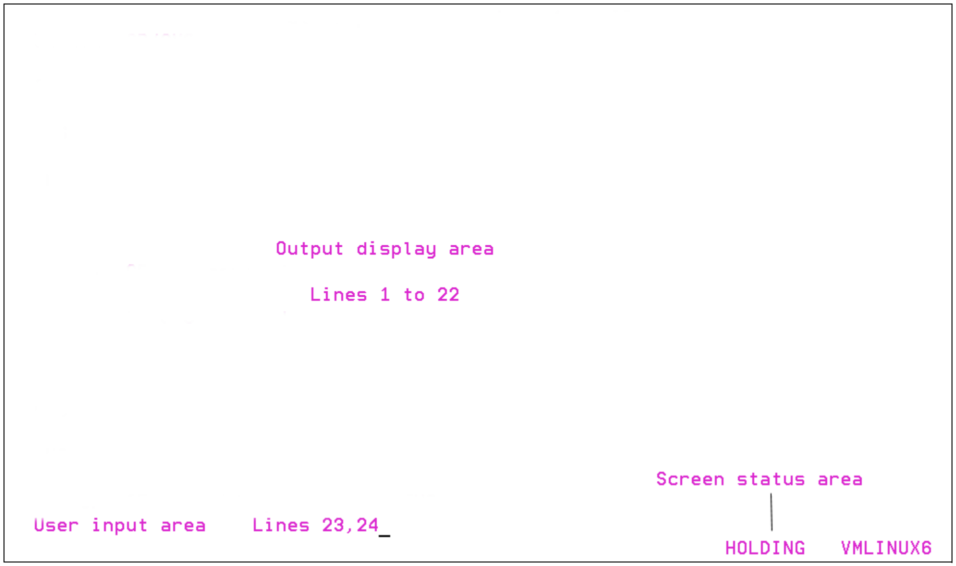

Figure 2-15 Layout of the 3270 screen

The screen is divided into three areas: the output display area, the user input area, and the screen status area, as described in Table 2-1.

Table 2-1 Screen display areas

Area on screen	Description
Output display area	► Consists of lines 1 to 22. ► All messages will be displayed here. ► The commands you entered will be displayed here.

<table>
<tr><th>Area on screen</th><th>Description</th></tr>
<tr><td>User input area</td><td>► Consists of last two lines, excluding the right-most 21 character positions of the last line.
► Commands can be entered here.
► Cursor control keys, DELETE key, INSERT Key and logical text editing characters can be used to alter data in this area.
► After pressing Enter, CP redisplays the data entered here in the output display area.</td></tr>
<tr><td>Screen status area</td><td>► Consists of the right-most 21 character positions of the last line.
► Screen status indicators appear here.
► Indicates whether the current environment is CP or VM.
► The status displayed here can be one of the following:
– CP READ - CP issued a read request to the display and is waiting for the user to enter something before it can continue processing.
– VM READ - The virtual machine is awaiting a response from the user before it can continue processing.
– RUNNING - CP or VM is either ready to accept commands, or processing an already executed command
– MORE ... - Indicates that the output display area is full and there is more data to display.[1]
• CP waits for 60 seconds (default) and then displays the next screen automatically.
• Use the PA2 key (3270) to see the next screen without waiting 60 seconds.
• Use the Enter key (3270) to keep the current screen (status changes to HOLDING).
– HOLDING - Indicates that user pressed Enter (3270) in response to MORE.
– NOT ACCEPTED - Indicates that the previous input was not accepted (CP locks keyboard for 3 seconds).</td></tr>
<tr><td></td><td>[1]Use SET FULLSCREEN ON from CMS if the scroll back option is needed. This is disabled by default. (Note that this works only for CMS.)</td></tr>
</table>

2.7.6 Session management

Your session begins when you log on to the system, and it ends when you log off. Here we describe the processes of logging in and logging out, as well as a few other actions you can perform during a session.

Logging on

For an overview of the process of logging on, see Figure 2-13 on page 59. Here, we explain how to log on in detail.

The logon screen contains three basic fields:

- Username
- Password
- Command line

If you want to log on normally, enter your user name and password in the Username and Password fields.

By using the command line, however, you can issue the CP **LOGON command**, which has many parameters that can customize how you log on to the system. To use the command line, press the Tab key until your cursor is positioned on the command line. (Or you can use the arrow keys or your mouse to manually move the cursor to the command line.)

After you are on the command line, issue the LOGON command. The simplest use of the LOGON command is shown in Example 2-4.

Example 2-4 Simple use of LOGON command

```
LOGON TUX1
```

This command will prompt you for a password. If the password that you enter is correct, the system will log on the user TUX1 just as if you had entered the user name and password in the normal Username and Password fields.

Notes:

- If you type a user name or password incorrectly on the login screen, then the normal input fields will disappear and you will be forced to use the LOGON command from the command line if you want to try to log in again. Therefore, it is important to know how to do this in case you make a typo the first time you try logging in.
- The CP **LOGIN** command is identical to the CP **LOGON** command. Both exist because some people prefer to use the term “log in” and others prefer “log on”.

Disconnecting

CP provides a very useful feature known as disconnecting. You may *disconnect* your terminal from your virtual machine without stopping or interfering with the guest operating system running in your virtual machine in any way. This effectively severs your actual terminal session with the virtual machine (like logging off), but the virtual machine and its guest operating system will continue to run. You disconnect by using the **DISC** command (for disconnect); see Example 2-5.

z/OS analogy: z/OS has no similar command

Example 2-5 Disconnecting from a guest

```
disc
DISCONNECT AT 11:34:15 EDT WEDNESDAY 06/13/07

Press enter or clear key to continue
```

You can use the **QUERY NAMES** command to tell which guests are disconnected. As shown in Example 2-6, disconnected guests have DSC displayed next to their names.

Example 2-6 Querying disconnected guests

```
   QUERY NAMES

   EDI      -L0005, DIRMAINT - DSC , TCPIP    - DSC , RSCS     - DSC
   PVM      - DSC , DATAMOVE - DSC , DTCVSW2  - DSC , DTCVSW1  - DSC
   VMSERVR  - DSC , VMSERVU  - DSC , VMSERVS  - DSC , GCS      - DSC
   OPERSYMP - DSC , DISKACNT - DSC , EREP     - DSC , OPERATOR - DSC
   CLIVE    -L0008, MAINT    -L0004, EDI2     -L0006, JASON    -L0007
   VSM     - TCPIP
```

When disconnecting from a guest, it is important to note your virtual machine's RUN parameter. If the RUN parameter is set to OFF when you disconnect (or reconnect), then your guest operating system may be suspended, in some cases.

To ensure that this does not occur, ensure that RUN is set to ON before you disconnect. You can do this by using the **SET** command as shown in Example 2-7.

Example 2-7 Setting the RUN parameter to ON

```
SET RUN ON
```

Reconnecting

To reconnect to a disconnected virtual machine, perform a normal logon and you will be reconnected. Once you are logged on you may have to reinitiate the session with your guest system by entering the **BEGIN** command.

Stealing a virtual machine session

If you ever find yourself in a situation where you need to log on to a virtual machine but it is alreadt logged on to another terminal, you will be presented with the output shown in Example 2-8.

Example 2-8 Logon failed because guest is already logged in

```
LOGON TUX1
HCPLGA054E Already logged on LDEV L0008
```

In this case, you may perform an action known as “stealing” the session by passing the **HERE** parameter to the **LOGON** command, as shown in Example 2-9.

Example 2-9 Stealing a virtual machine session

```
LOGON TUX1 HERE
RECONNECTED AT 11:48:17 EDT WEDNESDAY 06/13/07
```

As shown in Example 2-9 we have stolen the session from the other terminal connected to *TUX1*. What actually happens is that CP disconnects the other user session and then immediately connects to your terminal. Anything that was displayed on the screen or running before the disconnect will appear on your screen after the reconnect.

> **z/OS analogy:**This is similar to the TSO logon in z/OS with the RECONNECT parameter

Logging out

When you are finished running your virtual machine, you can log out by using the **LOGOFF** command. This will cause your virtual machine to terminate and it will be shtudown until you log on again. CP removes the virtual machine and it’s defined resources from it’s device list until the next time the virtual machine is logged on. CP. Logging off will immediately terminate any guest operating system that may be running.

2.7.7 Terminal Management

Your virtual terminal is represented by your terminal emulator. A *terminal emulator* is the program you are looking at when you are issuing commands to CP and reading the results. The terminal emulator typically presents a black background with mostly green text. CP provides several ways for you to modify what your terminal looks like and how it behaves.

Note: You can make persistent changes to your terminal by placing some of the commands shown in the next sections into you user's `PROFILE EXEC` file.

Setting the clear screen timeout

You may have noticed that when your screen fills up, CP displays the text **`More...`** in the lower right corner of the screen and waits for you to press the key on your keyboard that clears the screen before displaying more output.

By default, CP will wait 60 seconds for you to press the clear key before clearing it automatically. However, you can customize this behavior by using the **`TERM`** command:

```
TERM MORE t1 t2
```

Table 2-2 on page 67 lists the `TERM MORE` parameters.

Table 2-2 TERM MORE command parameters

Argument	Description
t1	This refers to the number of seconds that CP will wait before sounding a bell or beep to alert you to the fact that the screen is about to be auto-cleared for you.
t2	This refers to the number of seconds after the bell is sounded before the auto-clear actually takes place.

You can prevent CP from holding the screen and forcing you to press the clear key by setting both timeouts to zero (0) seconds, as shown here:

```
TERM MORE 0 0
```

Keep in mind, however, that by setting the timeouts to zero, you may not have time to read all of the messages presented to you on the terminal. In some cases, this may be acceptable (for example, if you are running a command or a program that is generating a significant amount of output that you do not care about).

You can return to the default behavior of 60 seconds (50 seconds before the bell, 10 additional seconds before the clear) by using this command:

```
TERM MORE 50 10
```

Note: The maximum value you can specify for both timeouts is 255 seconds. This means that the maximum wait before an auto-clear is 8 minutes and 30 seconds.

Highlighting user input

The CP **TERM** command has a parameter that causes all user input (that is, all of the text that you specifically enter) to be highlighted in a color different than the color used for other text output.

You can turn highlighting on by using the **TERM HILIGHT ON** command. Any command executed after this will be highlighted in a different color on the screen. This makes it particularly easy to visually separate input and output and generally makes your terminal text easier to read.

You can turn highlighting off by using the **TERM HILIGHT OFF** command.

Changing screen colors

You can change the colors that CP uses to display input, output, and status on your 3270 terminal emulator window by using the **SCREEN** command:

```
SCREEN area color effect
```

Table 2-3 lists the **SCREEN** command parameters.

Table 2-3 SCREEN command parameters

Argument	Description
area	This refers to the area of the screen you are interested in changing. Valid areas are listed in Table 2-4.
color	This refers to the color you want to use for the text in your chosen area. Valid colors are listed below.
effect	This refers to the effect you want to apply to the text in your chosen area. Valid effects are listed in Table 2-5.

Table 2-4 lists the valid areas for the **SCREEN** command.

Table 2-4 Valid areas for the SCREEN command

Area	Description
INAREA	The command line area where you issue commands to CP and guest operating systems.
STATAREA	The display of the terminal status at the bottom of the screen.
OUTAREA	The area where messages from CP and guest operating systems are placed for you to see. This is the area that takes up the vast majority of the screen.
CPOUT	This is not really an area. Rather, it is used to cause all messages that come from CP to be formatted as specified in the rest of the command.
VMOUT	This is also not really an area. Rather, it is used to cause all messages that come from a guest operating system to be formatted as specified in the rest of the command.

The following colors are valid for the **SCREEN** command:

- BLUE
- TURQUOISE
- RED
- PINK
- GREEN
- WHITE
- YELLOW

Table 2-5 lists the valid effects for the **SCREEN** command.

Table 2-5 Valid effects for the SCREEN command

Effect	Description
BLINK	This effect causes the text on the screen to blink.
REVVIDEO	Reverse video. This swaps the text color and the background color. For example, green text on a black background would appear as black text on a green background with the reverse video effect applied.
UNDERLIN	This causes the text to be underlined.
NONE	This means: apply no effects. Print the text normally in the color specified.

Note: To reset your terminal to the default state, use the following **SCREEN** command:

```
SCREEN ALL DEFAULT
```

2.8 Getting Started: The Basic Commands for z/VM

This section we discuss some of the basic commands to navigate around your z/VM system. For more details on the topics discuss here refer to the IBM Redbooks publication, *Introduction to the New MainFrame:* z/VM Basics.

2.8.1 CP

You will primarily interact with CP via a command line interface. You issue commands to CP by typing them on the command line and pressing *Enter* to submit them for processing.

CP commands are always between 1 and 12 characters in length, and are not case-sensitive. A command will often allow the specification of parameters, all of which are separated by a space (as in most command line environments).

Many commands also have abbreviations which are shorter to type but more cryptic. For example, the **QUERY NAMES** command can be shortened to **Q N**. Executing either of the two commands will accomplish the same task.

This chapter always uses the complete command and not the abbreviation. If you are interested in knowing the abbreviation for a command, refer to the **HELP command** or to *CP Commands and Utilities Reference*, SC24-6081.

Getting to CP mode

After logging in, any number of things could be going on, depending on the guest operating system your virtual machine is running. Because we are only interested in CP at this point, in our scenario we are going to shut down the guest operating system that is running and get into CP mode so we can issue commands directly to CP.

In your case, it is likely that your virtual machine will be running the CMS operating system. You can tell if you are running CMS by the output you get when you log in. If you see output similar to Example 2-10, then you are running CMS. The last line (starting with z/VM V5.3.0) is the first line of output from CMS.

Example 2-10 Output seen when running CMS at logon

```
z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0001 RDR,   NO PRT,   NO PUN
LOGON AT 09:54:43 EDT MONDAY 06/04/07
z/VM V5.3.0    2007-05-02 16:25
```

If no output appears you might still be running CMS, so enter the **QUERY CMSLEVEL** command to check further. If you see output similar to Example 2-11 on page 71, then you *are* running CMS.

Example 2-11 Output from QUERY CMSLEVEL command

```
QUERY CMSLEVEL
CMS Level 23, Service Level 701
Ready; T=0.01/0.01 09:57:17
```

Important: To stop CMS and enter into CP mode, you can issue the **#CP SYSTEM CLEAR** command. Note, however, that if you are *not* running CMS, then this command will still probably work—but it may damage your guest operating system. Issuing a **SYSTEM CLEAR** command is the equivalent of pulling your desktop personal computer's power cord out of the wall socket. It halts the system immediately and does not give the operating system a chance to finish what it is doing and perform cleanup tasks. And if the operating system is engaged in writing a file when this happens, you could lose the contents of the file or even permanently corrupt the file system.

Therefore, if you are *not* running CMS, then consult your guest operating system's manual for proper shutdown instructions.

z/OS analogy: For z/OS users this is like doing a SYSTEM RESET on the HMC for your LPAR

Example 2-12 Output from #CP SYSTEM CLEAR command

```
#CP SYSTEM CLEAR
Storage cleared - system reset.
```

This command clears your virtual machine's memory and stops and resets all of its virtual processors.

If you are truly in CP mode and not running a guest operating system, then you should be able to issue the **BEGIN** command and see the output displayed in Example 2-13.

Example 2-13 Output of BEGIN command when not running a guest operating system

```
BEGIN
HCPGIR453W CP entered; program interrupt loop
```

If different content is displayed, then chances are you are still running a guest operating system.

Examining your virtual machine

CP has many commands that involve getting information about your virtual machine and defining its virtual hardware configuration. Here we examine some of the commands you can use to get information about your virtual machine.

At this point, we focus on the **QUERY** command. **QUERY** will provide a great deal of useful information about your virtual machine and the system it is running on. It takes several arguments, most of which have sub-arguments of their own. We will only look at a subset of the capabilities of the query command.

z/OS analogy: Query is similar to the various DISPLAY commands in z/OS

CP version

The **QUERY CPLEVEL** command will tell you which version (version is commonly referred to as "level" in z/VM terminology) of CP you are running; see Example 2-14.

z/OS analogy: For z/OS, the similar information is displayed when system command **D IPLINFO** is issued. It could also be displayed from variables in TSO and ISPF.

Example 2-14 Output from QUERY CPLEVEL command

```
QUERY CPLEVEL
z/VM Version 5 Release 3.0, service level 0701 (64-bit)
Generated at 05/02/07 16:28:04 EDT
IPL at 05/03/07 13:06:26 EDT
```

Note: The **QUERY CPLEVEL** command is run when you log on to your virtual machine.

Your guest name

If you just logged in, then you probably know what your guest name is (a *guest name* is the uniquely identifying name given to your virtual machine). But in case you have more than one terminal open at the same time and forget which terminal goes with which guest, you can use the CP command QUERY USERID to find out the guest name; see Example 2-15.

z/OS analogy: In z/OS (when logged into ISPF), your user ID is normally displayed in the primary option panel

Example 2-15 Output from the QUERY USERID command

```
QUERY USERID
TUX1     AT VMLINUX6
```

The **QUERY USERID** command also tells you which “node” you are on. The term *node* refers to the instance of the z/VM operating system that your guest is running on. In this case, *VMLINUX6* is the name given to the z/VM instance that the *TUX1* guest is running on. When your system administrator installed z/VM, the administrator set the node name.

Note: The terms “virtual machine”, “guest”, and “user” all refer to a virtual machine. We use these terms interchangeably throughout this book.

Your virtual machine's resources

Your virtual machine (like any computer) is made up of CPU, memory, and devices. The **QUERY** command can show you basic information about your virtual machine, as well as information about these three different types of resources.

You can use the **QUERY VIRTUAL CPUS** command to display which virtual CPUs are defined for your virtual machine; see Example 2-16 on page 73.

Example 2-16 Output from the QUERY VIRTUAL CPUS command

```
QUERY VIRTUAL CPUS
CPU 00   ID  FF02991E20948000 (BASE) CP   CPUAFF ON
```

Example 2-16 on page 73 shows there is one virtual CPU. If you have more than one CPU, then your output will include an additional line for each additional CPU that you have.

You can use the **QUERY VIRTUAL STORAGE** command to display how much storage (or RAM) your virtual machine has. (z/VM refers to RAM as “storage”. We use the term storage to refer to RAM throughout this book, unless otherwise specified.)

The output in Example 2-17 on page 74 shows that our virtual machine has 32 megabytes of storage. The amount of storage may be given in Kilobytes (K), megabytes (M), gigabytes (G) or terabytes (T).

Example 2-17 Output from the QUERY VIRTUAL STORAGE command

```
QUERY VIRTUAL STORAGE
STORAGE = 32M
```

You can use the **QUERY VIRTUAL ALL** command to display a list of all of the devices on your virtual machine; see Example 2-18 on page 74.

Example 2-18 Output from the QUERY VIRTUAL ALL command

```
QUERY VIRTUAL ALL
STORAGE = 32M
XSTORE = none
CPU 00  ID  FF02991E20948000 (BASE) CP   CPUAFF ON
No AP Crypto Queues are available
CONS 0009 ON LDEV L0007   TERM STOP  HOST TCPIP    FROM 10.0.0.1
   0009 CL T NOCONT NOHOLD COPY 001    READY FORM STANDARD
   0009 TO TUX1      PRT DIST TUX1      FLASHC 000 DEST OFF
   0009 FLASH        CHAR       MDFY       0 FCB       LPP OFF
   0009 3215   NOEOF CLOSED    NOKEEP NOMSG NONAME
   0009 SUBCHANNEL = 0001
RDR  000C CL * NOCONT NOHOLD   EOF       READY
   000C 2540          CLOSED    NOKEEP NORESCAN  SUBCHANNEL = 0002
PUN  000D CL A NOCONT NOHOLD COPY 001    READY FORM STANDARD
   000D TO TUX1      PUN DIST TUX1      DEST OFF
   000D FLASH       000 CHAR       MDFY       0 FCB
   000D 2540   NOEOF CLOSED    NOKEEP NOMSG NONAME
   000D SUBCHANNEL = 0003
PRT  000E CL A NOCONT NOHOLD COPY 001    READY FORM STANDARD
   000E TO TUX1      PRT DIST TUX1      FLASHC 000 DEST OFF
   000E FLASH        CHAR       MDFY       0 FCB       LPP OFF
   000E 1403   NOEOF CLOSED    NOKEEP NOMSG NONAME
   000E SUBCHANNEL = 0004
DASD 0190 3390 LX6RES R/O 107 CYL ON DASD  CD31 SUBCHANNEL = 0005
DASD 0191 3390 DKCD37 R/W 005 CYL ON DASD  CD37 SUBCHANNEL = 0000
DASD 019D 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0006
DASD 019E 3390 LX6W01 R/O 250 CYL ON DASD  CD32 SUBCHANNEL = 0007
DASD 0401 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0009
DASD 0402 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0008
DASD 0405 3390 LX6W01 R/O 156 CYL ON DASD  CD32 SUBCHANNEL = 000A
```

Each virtual device in your virtual machine has a virtual device number (commonly referred to as a *virtual device address*) that is specific to your virtual machine. In Example 2-18 on page 74, 191 is the virtual device number for the minidisk on DASD pack (disk) labeled DKCD37. The address 0009 is the virtual console device that is used by your terminal session.

But the point here is to recognize that there are a variety of virtual devices, including some DASD packs, a punch card reader, and a printer. You may also have other devices, depending on how your virtual machine is defined.

Privilege classes

CP uses what are known as *privilege classes* to regulate the types of things that virtual machines are allowed to do.

> **z/OS analogy:** There is no concept of privilege class or similar in z/OS from the operating system perspective. This concept (or similar) would have to be assigned in RACF or a similar product. The Unix System Services component of z/OS, however, has permissions set in its file system.
>
> However privilege classes are conceptually similar to “permissions” in a Linux environments.

When your system administrator created your virtual machine, the administrator assigned you a certain subset of the privilege classes available on the system. Each privilege class is denoted by either a single letter or a single number, each one defining a certain role for the user having that class.

The IBM default privilege classes are A to G, and all but class G define certain system administrator roles. Class G defines the general user, and it is the default class for users of the system that have no administrative capability.

You can use the **QUERY PRIVCLASS** command to list the privilege classes for which your virtual machine is authorized; Example 2-19 on page 75 shows output from this command.

Example 2-19 Output from the QUERY PRIVCLASS command

```
QUERY PRIVCLASS
Privilege classes for user TUX1
Currently: G
Directory: G
```

This means that your virtual machine can execute any command that is valid for class G users. Because the set of commands in any given privilege class is customizable by the system administrator, the set of commands you are able to run may vary slightly from the example.

CP also provides a command, **QUERY COMMANDS**, that will list all of the commands available to you, so you do not need remember your privilege classes or which commands go with them. Example 2-20 on page 76 displays output from the command **QUERY COMMANDS**.

Example 2-20 Output from the command QUERY COMMANDS

```
QUERY COMMANDS
ADSTOP    ATTN       BEGIN     CHANGE    CLOSE
COMMANDS  COUPLE     CPFORMAT  CPU       DEFINE    DETACH
DIAL      DISCONNECT DISPLAY   DUMP      ECHO      EXTERNAL
FOR       INDICATE   IPL       LINK      LOADVFCB  LOCATEVM
LOGON     LOGOFF     MESSAGE   NOTREADY  ORDER     PURGE
QUERY     READY      REDEFINE  REQUEST   RESET     RESTART
REWIND    SCREEN     SEND      SET       SIGNAL    SILENTLY
SLEEP     SMSG       SPOOL     SPXTAPE   STOP      STORE
SYSTEM    TAG        TERMINAL  TRACE     TRANSFER  UNCOUPLE
UNDIAL    VDELETE    VINPUT    VMDUMP    XAUTOLOG  XSPOOL
DIAG00    DIAG08     DIAG0C    DIAG10    DIAG14    DIAG18
DIAG20    DIAG24     DIAG28    DIAG40    DIAG44    DIAG48
DIAG4C    DIAG54     DIAG58    DIAG5C    DIAG60    DIAG64
DIAG68    DIAG70     DIAG7C    DIAG88    DIAG8C    DIAG90
DIAG94    DIAG98     DIAG9C    DIAGA0    DIAGA4    DIAGA8
DIAGB0    DIAGB4     DIAGB8    DIAGBC    DIAGC8    DIAGD0
```

As you can see in Example 2-20, even for a class G user (which by default is the most restrictive permission class) there are many commands available. The entries starting with DIAG represent diagnose functions that your virtual machine can call. A *diagnose function* is used by programmers and is similar to a Windows API call or a Linux system call. You cannot execute these by specifying the name on the command line.

If you are logged on to a userid with Class B privileges, you can determine what privilege class is needed to execute a specific CP command by using the QUERY COMMANDS command with the specific command as an additional parameter.

Example 2-21 QUERY COMMANDS ATTACH output

```
q commands attach
ATTACH        IBMCLASS=B
Ready; T=0.01/0.01 15:09:46
```

2.8.2 CMS

Just like any operating environment, CMS has commands for working with files. Here is a list of the more common file management commands used by CMS users.

The FILELIST command

One of the most useful CMS commands, **FILEList** is intended to allow you to list what is stored on a minidisk and to look for files by yielding a list of matching results.

> **z/OS analogy:** Similar to z/OS ISPF option 3.4 with the exception that TSO/ISPF users don't have a file system as such. The user would have to know the data set **name** of the data set to be manipulated. The TSO command ISRDDN lists all data sets **allocated** to your TSO/ISPF session and will let you manipulate them

The command syntax is: `FILEList filename filetype fm ( options`

The first parameter, *filename*, is the file name. Only the files whose name match the file name supplied will be listed. The second parameter denotes the `filetype`[5], and the third one denotes the file mode[6] or directory ID. You will go through a more in-depth discussion about this triplet in 4.2.4, "Dealing with disks" on page 135. It is usually not that useful to list files for which you already know the name. Usually what you are willing to do is to find out which files are on a directory ID, or which files have the same name or type, or maybe you want to do all of these altogether. To accomplish that you can use the asterisk (*) as a commonplace or wildcard. So, to list all of the files in file mode A, you can just type **filel * * a**. The result of such a command is depicted in Figure 2-16. As you can see, the *A* file mode has only one file, the `PROFILE EXEC` file. Notice the triplet: `PROFILE EXEC A`.

```
CERON   FILELIST AO  V 169  Trunc=169 Size=1 Line=1 Col=1 Alt=0
Cmd Filename Filetype Fm Format Lrecl Records Blocks   Date     Time
    PROFILE  EXEC     A1      V    66       19      1 5/09/08 14:53:47






1= Help 2= Refresh 3= Quit 4= Sort(type) 5= Sort(date) 6= Sort(size)
7= Backward 8= Forward 9= FL /n 10= 11= XEDIT/LIST 12= Cursor

====>
                                                      X E D I T  1 File
```

Figure 2-16   File listing using the filelist command

[5] A filetype is known in systems like Linux to be the file extension.
[6] A file mode or directory ID is the disk on which the file is to be found.

You should notice here that a file is denoted by a triplet composed of file name, file type and file mode. Users of Linux and other modern systems would be familiar with the first two ones, which are analogous to the *filename.extension* scheme in those systems. The new paradigm here is the file mode itself.

> **z/OS analogy:** In z/OS the dataset name is like the filename in z/VM. z/OS best practices try to include description to the file type and owner of the file.

The file mode could be seen as the location of the file, basically denoting which disk it is on. This field is analogous to the path in systems like Linux, with the exception that it is composed of a single letter that denotes a disk. There is no concept of directories in the CMS file system.

Note also the first column of the **`FILEList command`** output which says `cmd`. You can actually input file management commands there, such as `erase`, and press Enter to have your command processed. This will save you some time when managing files.

When issuing commands, you will eventually forget to type in some optional parameters, such as the `replace` parameter to `copyfile` when you wish to overwrite a file, or request **`FILEList`** to list a file mode that does not exist within your user space. In those cases you will see an error message code followed by a short sentence that is eventually not enough to figure out what went wrong. You can, however, type **`help`** `error_code` at the CMS prompt, where `error_code` is the code of `the error you have just encountered. For example, if you type in` **`help dmslst069e`**, which is the error code for an inexistent file mode, you will get a help screen as depicted in Figure 2-17 on page 79.

```
 MSG DMSLST069E              All Help Information line 1 of 43
(c) Copyright IBM Corporation 1990, 2007

 DMS069E  <Output filemode|Filemode|Disk|Directory>
      <vdev|volid|mode((vdev)|dirname> (is) not accessed(;
       access_authority|access_authority)

Explanation:The specified disk, directory, or file mode has not been
accessed. If "Disk" is displayed and the disk is accessed, it may
not be correctly formatted for the command entered. (For example,
the command is trying to write a CMS-formatted file on an
OS-formatted disk.)

 For the CMSDESK command, a R/W disk with file mode A is needed.

 For the RECEIVE command, one of the following occurred:

   o A read-only file mode was specified on the RECEIVE command and
the file cannot be written onto this file mode.

   o RECEIVE attempted to read in a file sent using the DISK DUMP
command (or SENDFILE with the 'OLD' option), and in order to use
DISK LOAD to read the file in, file mode A must be accessed in
read/write mode.

PF1= 2= Top 3= Quit 4= Return 5= Clocate 6= ?
 PF7= Backward 8= Forward  9= PFkeys  10= 11= 12= Cursor

====>
                                                  Macro-read 1 File
```

Figure 2-17 Getting help out of an error code

The COPYFILE command

The **COPY** command copies a file from one location to another. Simply specify the source and destination filenames file extensions and file modes. Though this might sound a bit trivial, it is very important to the CMS user.

Typically, a CMS user has read access allowing them to look at, and list, files that are on disks that are read only. In order to change those files, a CMS user must first copy them to a read-write disk. The command which allows you to do this is the **COPYFILE command**, as illustrated here:

```
COPYfile fileid1 (fileid2 .. fileidn) ( options
```

A simple example is to issue **`copyfile duane txt a secret data d`**. This makes a copy of the file *duane txt* on your A disk and stores it as "*secret data*" on your *D* disk.

However, you do not always have a *D* disk, or it may not always allow you to write files onto it. In such cases, you will receive an appropriate error message. (Note that the original file will not be not erased.)

The **`COPYFILE command`** is one of the most powerful CMS commands. In addition to the trivial copying operations described, this command allows you to specify multiple input files, copy specific columns to new positions, insert fixed data in the file, select out ranges of records to copy, and perform other actions. The command **`COPY MARK TXT S = = A`** will copy the file **`MARK TXT`** from the system (or S) disk to your A disk.

Another example invocation of the command is **`copyfile duck list c finch = = warbler = a birds file a (append`**. This example combines the files duck list and finch list on your C disk, and the file warbler list on your A disk, and adds the new combined file to the end of the file birds file a, which already exists. This form of the command is similar to using the **`cat`** and **`>>`** combination, for those familiar with Linux operating systems.

The **`COPYFILE command`** has additional useful options. Here is an example command that copies files from your S disk to your A disk, while ensuring that the time stamp on the system file is preserved:

```
COPY ALL XEDIT S = = A (OLDDATE
FILE 'ALL XEDIT A2' ALREADY EXISTS -- SPECIFY 'REPLACE'.
```

You will note that this example illustrates a common error that occurs when the output fileid already exists (for instance, after a previous COPY operation where the user neglected to keep the time stamps). In such a situation, you must reissue the COPY command with another option, as shown:

```
COPY ALL XEDIT S = = A (OLDDATE REPLACE
```

The open parenthesis (in all these examples is a CMS convention, and things that change the default operation of a command (that is, options) are placed after it. Note that you do not need to add the closing parenthesis).

> **z/OS analogy:** An analogy in z/OS would be to allocate the target file using ISPF option 3.2 prior to copying using option 3.3. It would also be possible to create a copy of a file from within a ISPF EDIT session.

The ERASE command

The **ERASE** command removes files from disk storage, releasing space for other uses. It gets rid of a file, and the space that the file occupied is immediately reusable. The syntax of the **ERASE** command is shown:

```
ERASE fn ft (fm) ( options
```

An example use of the **ERASE** command is **erase frank tdata a**. This would delete the file *frank tdata* on your *A* disk.

> **Important:** This command provides an immediate, irrevocable deletion of the file data. **ERASE** will assume that the file is on your A disk unless you specify otherwise.

2.8.3 XEDIT

You have now learned how to examine your accessed disks, and work with files at the file system level. But what about editing at the file level? Creating and changing content is an important part of the CMS users skill set. This section will introduce you to the utility most commonly suited for editing files on a CMS system.

XEDIT is the name of the CMS editor, and is one of the most flexible and powerful utilities provided with CMS. The shortest abbreviated command used to edit a file with xedit is **X** followed by the file name, for example **X PROFILE EXEC**. The full command invocation of the same example would be **XEDIT PROFILE EXEC A**.

> **z/OS analogy:** The z/OS equivalent is ISPF edit

The basic XEDIT screen displays several areas: a command area, an edit area. a prefix area, and a line at the top of the screen that gives information about the file being edited.

The command region is a single line at the bottom of the terminal. Here you can issue commands such as **SAVE**, **FILE**, and **QUIT**. If you type anything into the edit region (most of the rest of the screen), you will be modifying the loaded file. This can frustrate new CMS users who are not used to command and data being manipulated the same way.

> **Important:** Pressing Enter at any time will bring your cursor directly to the command line, but any text already on the command line will be executed!

Editing means changing, adding, or deleting data in a CMS file. You make these changes interactively; that is, you instruct the editor to make a change, the editor makes it, and then you request another change.

The XEDIT window layout

This sections describes the various areas of the XEDIT window, and explains what each are is used for.

File identification line

The file identification line (which is the first line on the screen) identifies the file you are editing. The display shows information about the file name, file type, file mode of the current file.

If you do not specify a file mode, the editor assigns a file mode of A1. The file mode identifies an accessed minidisk or SFS, for Shared File System, directory where the file resides.

The record format and record length (V 132) indicate the maximum length of line (the integer portion) and the V indicates lines may be variable length. Note that it is possible for a file line to be longer than a screen line.

The truncation column is the same as the record length (132). Because a file line can be only 132 characters long, any data you enter beyond 132 characters (in total) can be truncated. Additionally, you can see information about the current number of lines in the file. Finally, notice the alteration count shown. This value indicates the number of lines modified since the last AUTOSAVE.

Message line

The editor communicates with you by displaying messages on the second and third lines (the message lines) of the screen. These messages tell you if you have made an error, or they provide information.

Edit area

The file area part of the screen is available to display the file. You can make changes to the file by moving the cursor to any line and typing over the characters, or by using special keys to insert or delete characters.

You can make as many changes as you want on the displayed lines before pressing Enter. When you press Enter, the changes are made to the copy of the file that is kept in virtual storage.

At the end of the editing session, a **`FILE`** subcommand permanently records those changes on the copy of the file that resides on disk or directory. Because a file can be too long to fit on one screen, various subcommands scroll the screen so you can move forward and backward in a file. Scrolling the screen is like turning the pages of a book.

Prefix area

The prefix area consists of the five left-most columns on the screen. Note that each line in the file has a prefix area.

The area may display five equal signs (=====), or sometimes line numbers, depending on the particular configuration of XEDIT being used. In some cases the prefix area has no special marker at all.

You can perform various editing tasks, such as deleting a line, by entering short commands called *prefix subcommands in the prefix area* of a line.

z/OS Analogy: The prefix sub commands in XEDIT are the equivalent of line commands in the ISPF editor

The current line

The current line is the file line in the middle of the screen (above the scale). It is highlighted, appearing brighter than the other file lines. The current line is an important concept, because most subcommands perform their functions starting with the current line.

The line that is current changes during an editing session as you scroll the screen, move up and down, and so forth. When the current line changes, the line pointer (not visible on the screen) has moved. Many XEDIT subcommands perform their functions starting with the current line and move the line pointer when they are finished.

Note: Sometimes XEDIT is configured to show a scale that appears under the current line to help you edit. It is like the margin scale on a graphical text editors.

Command area

The large arrow (====>) displayed at the bottom of the screen points to the command input area. One way you communicate with the editor is to enter XEDIT subcommands on this line. You can type subcommands in upper case or lower case or a combination of both, and many can be abbreviated. For example, `BOTTOM`, `Bottom`, and `b` are all valid formats for entering the BOTTOM subcommand.

After typing a subcommand on the command line, press Enter to execute the subcommand. To move the cursor from any place on the screen to the command line, simply press Enter or PF12)

Status area

The lower right corner, the status area, displays the current status of your editing session (for example, edit mode or input mode), and the number of files you are editing.

z/OS analogy: In z/OS, the ISPF editor has no status line of this type

XEDIT and full screen CMS

If you invoke XEDIT from full screen CMS, the way you see messages that other users send you is not the same as when full screen CMS is off.

- When full screen CMS is off, the message appears on a cleared screen with a `HOLDING` status displayed at the bottom. You can press Clear to return to the XEDIT screen.
- If full screen CMS is on, then any message you receive appears in the message window, which automatically pops up on top of your XEDIT screen.

  To scroll forward in the message window, type `f` (forward) in one of the border corners (indicated by plus (+) signs) and press Enter.

  Continue to use the f border command until you have seen all the information in the message window. When there is no more information to display, the window is automatically removed from your screen.

Data manipulation with prefix subcommands

After you enter the XEDIT command, you are in typically in edit mode. You must be in edit mode to enter XEDIT subcommands. You can enter data into the file using input mode or power typing mode, which are discussed in the following sections.

If you are editing an existing file, you can simply place the cursor in the file area and type to replace the underlying text. As previously mentioned, XEDIT refers to its “shortcuts” as prefix subcommands. Prefix subcommands are one- or two-character commands that perform basic editing tasks on a particular line.

You enter prefix subcommands by typing over any position of the five-character prefix area on one or more lines. When you press Enter, all of the prefix subcommands that have been typed on the screen are executed.

z/OS analogy: XEDIT prefix subcommands are similar in function to ***line commands*** in ISPF editor.

Setting the current line

Many subcommands begin their operations starting with the current line. For example, the **INPUT** subcommand makes room for you to enter data after the current line. You have already seen the **INPUT** subcommand that inserts lines after the Top of File line.

You can type the forward slash (/) prefix subcommand in the prefix area of any line on the screen. When you press Enter, that line becomes the current line. Then, if you enter an **INPUT** subcommand, the new lines entered in input mode are inserted between the current line and the line that followed it.

Adding lines

To add a line, type the single character `A` (or the character `I` for insert) in the prefix area to append blank lines. When you press Enter, a blank line is immediately inserted following the line containing the A. A number can precede or follow the A to indicate adding more than one line. For example, A5 adds five blank lines.

Here are valid ways to type the A prefix subcommand:

```
====A     Adds   one blank line after this line.
a====     Adds   one blank line after this line.
10a==     Adds   ten blank lines after this line.
===A5     Adds   five blank lines after this line.
```

You can then type information in the added lines. If no information is typed, the blank lines remain in the file throughout the editing session and after the file is written to disk or directory.

z/OS analogy: In ISPF EDIT, you would typically use the I(nsert) line command.

Deleting lines

To delete a line, enter the single character D in the prefix area of a line. A number can precede or follow the D to indicate deleting more than one line.

To delete a group of consecutive lines (that is, a block of lines) enter the double character DD in the prefix area of both the first and last lines to be deleted. This method makes it unnecessary for you to count the number of lines to be deleted; see Example 2-22.

Example 2-22 The consecutive line delete functionality

```
==dd=  This   is  the first line I want to remove.
=====  This   is  the second.
=====  This   is  the third.
=====  This   is  the fourth.
===dd  This   is  the fifth.
```

When you press Enter, the block of lines is deleted. The first and last lines of the block do not need to be on the same screen; you can scroll the screen before entering the second DD.

When you have typed one `DD` and pressed Enter, the status area of the screen displays `DD pending...`. You can use the PF7 or PF8 keys to scroll the screen until you find the last line of the block, and then type `DD` in its prefix area. When you press Enter, the entire block of lines is deleted.

z/OS analogy: Functions identical in both editors

Recovering deleted lines

If you delete one or more lines, you can recover them anytime during an editing session by using the **RECOVER** subcommand. The following subcommand returns lines deleted in an editing session:

RECover n

Here, n represents the number of lines you wish to recover. Recovered lines are inserted starting at the current line. The last lines deleted are the first lines recovered.

If the lines were deleted from different places in the file, you put them back where they belong by using the **M** prefix subcommand (refer to “Moving lines” on page 88 for more information about this subcommand). To recover all lines that you deleted during an editing session, enter:

====> **recover**

In the previous example of the `A` and `D` prefix subcommands, six lines were deleted. To recover only the last 2 lines, use the following command:

====> **recover 2**

z/OS analogy:: Could be similar to using RECOVERY ON command in ISPF editor and typing UNDO, provided you have pressed ENTER between each editing step you wish to recover.

Adding indented lines

To continuously add lines of indented text, type the characters SI in the prefix area. When you press Enter, a line is immediately added following the line that contains SI. The cursor is positioned at the same column where the text on the previous line begins, thus making it easier for you to enter indented text.

If you do not want to add more lines, press Enter one more time without typing anything on the new line. To add a blank line in a file while using SI, make at least one change (such as pressing the spacebar once) on the line that contains `. . . . .` in the prefix area. Note that simply using the cursor position keys to move the cursor over a line does *not* change the line.

You can leave the line you are adding and make corrections elsewhere in the file if you type something on the new line first. When you press Enter while the cursor is away from the new line, another new line is added following the last line that was added. SI is canceled only if you press Enter and have typed no text on the new line.

Duplicating lines

To duplicate a line, enter a double quote (") in the prefix area of a line. A number can precede or follow the double quote to duplicate the line more than one time; see Example 2-23.

Example 2-23 Inserting three duplicate lines

```
=3"== I want three more copies of this line.
===== They will appear before this line.
```

When you press Enter, the file displays output similar to Example 2-24.

Example 2-24 Output from the previous example

```
=====    I want three more copies of this line.
=====    I want three more copies of this line.
=====    I want three more copies of this line.
=====    I want three more copies of this line.
=====    They will appear before this line.
```

To duplicate a block of lines, either one time or a specified number of times, type two double quotes ("") in the *first* and *last* lines of the block. A number can precede or follow the first double quotes (for example, 5"") to duplicate the block more than one time.

When you type one double quote ("") and press Enter, the status area of the screen displays `"" pending...`. This allows you to scroll the screen before completing the block and pressing Enter.

z/OS analogy: In z/OS, similar to using R(epeat) n line command in ISPF editor

Moving lines

To move one line, enter the single character M in the prefix area of the line to move. Indicate its destination by entering either the character `F` (following) or `P` (preceding) in the prefix area of another line. When you press Enter, the line containing the M is removed from its original location and is inserted in one of the locations (either immediately following the line containing the F, or immediately preceding the line containing the P, as appropriate).

A number can precede or follow the M to indicate moving more than one line (for example, 5M or M5) in the prefix area. The line to move and the destination line can be on different screens.

After you enter M, F, or P, the status area of the screen displays a `Pending` status. This pending status allows you to scroll the screen before entering the other prefix subcommand.

To move a block of lines, enter the double character MM in the prefix area of both the *first* and *last* lines to be moved. The first and last lines to be moved, and the destination line, can all be on different screens. You can use PF keys to scroll the screen before pressing Enter.

z/OS analogy: In z/OS (ISPF editor) the **M**(ove) command is identical, but line command **A**(fter) is used instead of **F**(ollowing), and **B**(efore) instead of **P**(receeding)

Copying lines

The procedure for copying lines is the same as for moving lines, except that you enter a `C` or `CC` prefix subcommand instead of M or MM. The copy operation leaves the original lines in place but makes a copy at the destination line, which is indicated by F or P.

z/OS analogy: In z/OS (ISPF editor), the **C**(opy) command is identical, with the same exceptions as stated for the M(ove) command previosly

Canceling prefix subcommands

If you have entered one or more prefix subcommands that create a pending status, you can cancel all these prefix subcommands by entering the following subcommand on the command line:

```
====> reset
```

When you press Enter, all prefix subcommands disappear from the display and the prefix areas are restored with equal signs, numbers, or blank characters, depending on your particular XEDIT configuration.

If you have typed any prefix subcommands (even those that do not cause a pending status) but have not yet pressed Enter, you can press Clear to remove them. If you have only typed a few characters, it may also be sufficient to just type over them with blank characters.

Moving through a file

XEDIT lets you move backward, forward, to the top and bottom, and up and down in a file. You have already seen that the PF7 and PF8 keys are set to the BACKWARD and FORWARD subcommands, which scroll one full screen backward or forward.

You can also enter the BACKWARD and FORWARD subcommands in the command line. The format of these subcommands is:

```
BAckwardn
FOrwardn
```

Here n is the number of screen displays you want to scroll backward or forward. (This is like pressing PF7 or PF8 n times.) If you omit n, the editor scrolls one screen backward or forward.

If you enter a BACKWARD subcommand when the current line is the Top of File line, the editor wraps around the file, making the last line of the file the new current line. Similarly, if you enter a FORWARD subcommand when the current line is the End of File line, the editor makes the first line of the file the new current line.

Suppose the file is many screens long and the current screen display is somewhere in the middle of the file. To return to the beginning of the file, you could enter multiple BACKWARD subcommands, or you can enter the **TOP** subcommand.

The TOP subcommand makes the Top of File line the new current line. Enter the TOP subcommand as shown:

```
====> top
```

The BOTTOM subcommand makes the last line of the file the new current line. Enter the BOTTOM subcommand as shown:

```
====> bottom
```

These subcommands are useful when you want to insert new lines either at the beginning or end of a file.

Suppose that you want to move the file up or down a few lines instead of a whole screen. The **DOWN** subcommand advances the line pointer one or more lines toward the end of a file. The line pointed to becomes the new current line; for example:

```
====> down 5
```

This command makes the fifth line down from the current line the new current line. If you omit the number, then 1 is assumed. The **UP** subcommand moves the line pointer toward the beginning of the file. The line pointed to becomes the new current line; for example:

```
====> up 5
```

This command makes the fifth line up from the current line the new current line. If you omit the number, then 1 is assumed.

z/OS analogy: In z/OS (ISPF editor), these commands are identical

Searching within a file

Searching for a particular word or phrase is a very important part of text editing on any platform, and is especially important when using CMS to edit various configuration files. When using XEDIT, you can search by using the forward slash (/) command on the command line, as shown:

```
====> /target_search_string
```

After pressing return, XEDIT will move to the first line it has found that matches your search phrase. XEDIT searches by default, moving from the current line toward the end of the file. To move to the next occurrence, you will have to reenter the command:

```
====> /target_search_string
```

A more convenient solution is to prefix the search command with the & character, as shown:

```
====> &/target_search_string
```

After pressing return, XEDIT will move to the first line it has found that matches your search phrase. Each time you press return again, XEDIT will move to the next match in the file.

This can be a very powerful way to move through files, but you can also search in reverse. To search in reverse, prefix the search command with the minus (-) symbol. The search will be performed from the current line moving toward the top of the file. Here is an example of a reverse search:

```
====> -/target (reverse search)
```

Lastly, in some cases highlighting all occurrences of a phrase within a file can be useful. To accomplish this task, XEDIT provides the **`all`** command:

```
====> all /target
```

From the XEDIT command line, the **`all`** command will show a listing of all lines that contain instances of the target search phrase. To return to normal editing mode, issue the all command with no search phrase as shown:

```
====> all
```

> **z/OS analogy:** In z/OS this is similar to **F(ind) 'txt'** primary command of ISPF editor

> **Note:** When using the **`all`** command or the search / command, you can set your current line to be one of the lines of output. This will enable you to exit the search results and be immediately editing the file at your position of interest.

Setting tabs

You may want to place information in specific columns. The PF4 key functions like a tab key on a typewriter. Each time you press the PF4 key, the cursor is positioned under the next tab column, where you can enter data. The editor defines initial tab settings according to file type; you can display them with the following subcommand:

```
====> query tabs
```

You can change these settings one or more times during an editing session with the SET TABS subcommand; here is an example:

```
====> set tabs 10   20   30
```

The first time you press PF4, the cursor moves to column 10 on the screen. The second time, it moves to column 20, and so forth. You can use PF4 for tabbing in regular XEDIT input mode, but not all other advanced xedit modes. You can change the tab settings by entering another **`SET TABS`** subcommand. If you want to see the current tab settings before changing them, use the following subcommand:

```
====> modify tabs
```

This displays the current SET TABS subcommand in the command line; you can type over the numbers and press Enter to define new tabs.

Inserting from external files

The **GET** subcommand inserts all or part of another file into the file you are editing after the current line. (A file that you “get” is not destroyed; a copy of that file is inserted.)

Before you enter the **GET** subcommand, make the current line the line *preceding* where you want to insert data. That way, the file will be inserted immediately under the current line.

To insert another file at the end of your file, use the **BOTTOM** subcommand to make the last line current. To insert another file somewhere in the middle of your file, use the **UP** or **DOWN** subcommands to make the desired line current.

z/OS analogy: Similar to **COPY** primary command of ISPF editor in z/OS

Inserting a whole file

To insert all of another file into the file you are editing, use the command:

```
====> get filename filetype
```

For example, to insert all of FILE2 SCRIPT A at the end of FILE1, then while you are editing FILE1, move the line pointer to the end of the file by issuing the command:

```
====> bottom
```

With the line pointer at the end of our current file, import the entire external file with the command:

```
====> get file2 script
```

When the entire second file has been inserted, the editor displays the message:

```
EOF reached
```

Inserting a portion of another file

Sometimes importing an entire file is unnecessary; instead, you only need a portion of an external file. The **GET** command takes additional optional arguments that will help you to insert a portion of another file.

The first additional parameter for this invocation specifies the line number of the first line to import. The second additional parameter indicates the number of suasive lines to insert. For example, the following **GET** subcommand inserts the first 10 lines of a second file:

```
====> get file2 data 1 10
```

Powertyping

If you need to type a significant amount of text continuously, without worrying about line numbers or word length, enter the command **POWERINP**. This places XEDIT in a mode called power input or powertyping mode.The shortest command abbreviation is **POW**.

When you are in powertyping mode, a `*** Power Typing***` banner is displayed across the top of the screen. Any time you want to leave powertyping mode, simply press the return key. Your text will be formatted to fit the correct line width in a normal XEDIT session.

Combining the **TOP** or **BOTTOM** commands with **POWERINP** provides an easy method for prepending or appending to a file.

z/OS analogy: The ISPF editor of z/OS has no power typing abilities

AUTO SAVE

To minimize the risk of losing your data, XEDIT provides the **SET AUTOSAVE** subcommand. This subcommand causes your file to be automatically written to disk after you have typed in or changed a certain number of lines. Its format is:

```
SETAUtosave n
```

Here n is the number of typed-in or changed lines. For example, to write the file to disk or SFS directory every time you have changed 10 lines, enter:

```
====> set autosave 10
```

The number of alterations you have made to your file since the last AUTOSAVE is displayed in the alteration count (Alt=n) in the file identification line. When the alteration count is equal to the AUTOSAVE setting, and the file contains at least one record, the file is saved on disk or SFS directory and the alteration count is reset to zero (0).

You can enter the **SET AUTOSAVE** subcommand at any time during an editing session, but it is good practice to enter it right after you enter an XEDIT command to create a new file or to call an existing file from disk or SFS directory.

When a file is automatically saved, it is written into a new file whose file name is a number and whose file type is AUTOSAVE. If the system malfunctions during an editing session, you can recover all changes made up to the time of the last automatic save.

To do this, replace the original file with the AUTOSAVE file using the CMS **`COPYFILE command`** with the **`REPLACE`** option. If you enter a **`SET AUTOSAVE`** subcommand while you are creating a new file or revising an existing file, and then enter a **`QUIT`** subcommand, the new or revised file is not saved, but the AUTOSAVE file is available from disk.

z/OS analogy: The ISPF editor has not got this ability. An EDIT profile value of AUTOSAVE ON in ISPF implies that exiting the editor would save the data set or member if it has been modified.

If AUTOSAVE is set to NO, the editor will prompt the user to enter SAVE or CANCEL (very similar to XEDIT)

Ending an editing session

You can end an editing session by using **`FILE`** or **`QUIT`**. When you use the **`XEDIT`** command to create a new file, the file is created in virtual storage. When you make changes to an existing file, those changes are made to a copy of the file that is brought into virtual storage (when the **`XEDIT`** command is entered). However, virtual storage is temporary. To write a new or modified file to disk, enter the following subcommand:

```
====> file
```

When the **`FILE`** subcommand is executed, the file is written to disk or directory and control is returned to CMS. You must use **`FFILE`** to file an empty file.

The **`QUIT`** subcommand ends an editing session and leaves the permanent copy of the file intact on the disk or directory. You can execute the **`QUIT`** subcommand either by pressing the *PF3* key or by entering it on the command line, like this:

```
====> quit
```

Use the **`QUIT`** subcommand to quit XEDIT without saving changes to the file. This method of exiting XEDIT may be useful when you edit a file just to examine it but not change its contents, or if you have made an error when altering a file.

If the file is new and you have not input any data, the file is not written to disk. If a file is new or has been changed, the editor gives you a warning message to prevent your inadvertently using **`QUIT`** instead of **`FILE`**. The message is:

```
File has been changed; type QQUIT to quit anyway
```

If you really do not want to save the file, enter **QQUIT** (abbreviated as **QQ**):

```
====> qquit
```

z/OS analogy: z/OS/ISPF editor analogy: Primary command CAN(cel) is identical to exit without saving ... SAVE saves the data set or member regardless if it has changed or not. Pressing PF3 saves if data is changed.

Customizing XEDIT

You can improve the X interface by creating a PROFILE XEDIT file on your A disk. XEDIT starts with the basic requirements of any data editing program (that is, it should allow you to add, delete, or change records in a file interactively from your screen). It also provides a significant range of additional and more specialized functions. Some of the advanced features are:

- Sophisticated data location commands
- Program controllable changes
- Display layout “tailoring”
- Selective data display
- Multiple files handled simultaneously
- Multiple logical displays per physical display

Recognizing the variety of different users who will need an editor, and their different expectations and requirements, XEDIT allows each user to define how file data should be displayed and how commands should take effect. Discussing these advanced features, and many more, are beyond the scope of this book.

A sample PROFILE XEDIT is shown in Example 2-25.

Example 2-25 Sample PROFILE XEDIT A with basic XEDIT customization

```
* * * Top of File * * *
/****************************/
/*   Sample PROFILE XEDIt   */
/****************************/

/* Set up function keys to do something useful */
'SET PF01 HELP MENU'    /* XEDIT help                                */
'SET PF02 SOS LINEADD'  /* Add a line at cursor position             */
'SET PF03 QUIT'         /* Quit XEDIt                                */
'SET PF07 BACKWARD'     /* Scroll backward                           */
'SET PF08 FORWARD'      /* Scroll forward                            */
'SET PF09 ='            /* Re-execute last subcommand entered        */
'SET PF10 RIGHT 10'     /* Scroll document to right 10 columns       */
'SET PF11 LEFT 10'      /* Scroll document to left 10 columns        */
'SET PF12 ?'            /* Retrieve last command issued.             */

/* Set up colors */
```

```
'SET COLOR PREFIX BLUE'
'SET COLOR ARROW WHITE'
'SET COLOR FILEAREA GREEN'

/* Set up display of editing window */
'SET NUM ON'         /* Show line numbers                             */
'SET CMDLINE BOTTOM' /* Command line at bottom of screen              */
'SET SCALE OFF'      /* No "scale" bar across window                  */
'SET NULLS ON'       /* Allows you to insert text in middle of a line */
'SET CASE M I'       /* Allows uppercase and lowercase characters     */
'SET CURLINE ON 3'   /* Current line is always the 3rd visible line.  */
'SET FULLREAD ON'    /* move cursor to any spot and char stays put    */
'SET STAY ON'        /* stay at loc of last find vs bottom of file    */
* * * End of File * * *
```

Experiment with the various option in the configuration file. For example, saving the file to PROFILE XEDIT A in your CMS system will put the configuration overrides into production. It is recommended that you start by adding one line at a time to determine what you want. The configuration shown in Example 2-25 on page 95 may not meet your needs, but it is presented to illustrate the customization available to XEDIT users.

z/OS analogy: The ISPF **EDIT** profile works in a similar way.

Tip: The XEDIT PROFILE lets you customize XEDIT so it behaves more similar to ISPF EDIT if desired. The SYNONYM command is a good place to start. See z/VM: XEDIT User's Guide, SC24-6132 manual

Getting help with XEDIT

If you forget how to use a subcommand, or want to view information about subcommands not covered in this subset, press *PF1*, which is set to the **HELP MENU** subcommand. *PF1* lists all subcommands and macros available with the editor.

If *PF1* does not display a help menu for XEDIT, you can manually enter the command HELP XEDIT MENU. Move the cursor to the desired subcommand and press Enter. The subcommand description appears on the screen, replacing the Full-Screen Text Processing HELP Menu.

To return to the previous screen, press *PF3*. To leave the HELP display and restore your file on the screen, press *PF4*.

z/OS analogy: In the ISPF editor, Pressing PF1 from ISPF editor work in a similar way, but could not be used for EDIT primary or line commands directly. Instead, they are presented in hiearchically organized panels.

3

Introducing Linux on System z to z/OS system programmers

In this chapter we give an overview of where Linux came from, where it is now, and where it is going. We also take a deeper look into Linux on System z and the benefits from it

.

Objectives

On completion of this chapter you should be able to:

- Understand basic concepts of Linux
- Have basic skills to navigate around Linux
- Have general knowledge of basic Linux commands

3.1 Linux Overview

This section gives an overivew of Linux and how it interacts with the System z. Plus, it looks at what comprises a distribution of Linux on System z, and highlights the major features of the most popular enterprise Linux editions for System z distribution.

3.1.1 History of Linux

In the simplest terms, Linux is an operating system. It was created in October 1991 by a University of Helsinki student named Linus Torvalds (Linux stands for Linus's UNIX). Linux itself is actually just the kernel; it implements multitasking and multiuser functionality, manages hardware, allocates memory, and enables applications to run.

Over the past few years, the Linux operating system has become a real and viable alternative for PC users as well as corporate servers and users. Linux delivers the power and flexibility of a UNIX server or desktop. It also provides a set of utilities, Internet applications, and a fully functional desktop interface.

The Linux operating system has become a server platform for powerful Internet and many other applications. Linux is capable of running from corporate Web, File Transfer Protocol (FTP), file and printer servers to wide-area information server (WAIS) Web sites or even corporate database servers, with real-time clusters or high performance computing clusters.

Linux is a fully functional operating system similar to a UNIX system. It has many of the standard features of enterprise UNIX systems. Management of the command structure is enabled through a *shell.*

z/OS analogy: This shell is much like the ISPF shell (ishell).

Enhanced graphical environments and desktops are also available.

The structure of Linux is organized on file systems that provide the interfaces and abstraction needed to work with data and files. Files are organized into directories with the disk hardware. Each directory may contain any number of subdirectories each holding files. This is a similar structure to classical PC and UNIX operating system file structures. Linux supports the majority of the existing file systems, either by having kernel built-in support or through kernel modules.

3.2 Components of Linux

Linux has three fundamental components:

1. Kernel

This provides low-level system control and interfaces, program and hardware device management, and an abstraction layer for the user level.

2. init

This is usually the main program executed after the kernel is loaded into memory. From init, you can execute different run levels on the user space, with different characteristics.

3. Applications

Linux applications cover many categories. The list is ever-expanding as more companies embrace this technology. Through writing applications for Linux, development costs usually decrease, spawning new projects and increasing the number of titles overall. Across the many horizontal and vertical markets with a Linux presence, we can cite eleven categories: office productivity, Internet related, network and systems management, software development to video production, data management, accounting and finance, and publishing.

Linux has the same multi-user and multitasking capabilities as large UNIX operating systems. It provides the same level of system administration that you find on standard UNIX systems. Users can run multiple programs concurrently, and there is a separate level for user space than kernel space. You can create user accounts for different users, and define their access rights to the files and system components. Installation of new devices and network connection and control is also provided as standard in the Linux operating system.

3.3 Linux on System z distributions

Supported Linux distributions for System z are currently available from two major Linux distributors: Novell's SuSE Linux Enterprise Server (SLES) and Red Hat Enterprise Linux (RHEL). Each distribution of Linux has its pros and cons.

Linux supports the S/390 architecture has been generally available since early 2000 either from IBM developerWorks® or from the Linux distribution partners. All the current distributions (as of May 2008) are based on the Linux 2.6 kernel. Each distribution includes additional middleware and applications. They also include infrastructure services such as domain name server (DNS), Dynamic

Host Configuration Protocol (DHCP), and Network File System (NFS) file servers, and packages such as the Apache Web server, Squid proxy server, Simple Mail Transfer Protocol (SMTP) mail server, and Samba Windows networking server. These distributions also leverage the HiperSockets technology of the System z server to interconnect between different partitions.

The distributions that are available for System z are:

- Novell SuSE
  - SUSE® Linux Enterprise Server (SLES) 10, Kernel 2.6.16
    - http://www.novell.com/partners/ibm/mainframe/
- Red Hat RHEL
  - Red Hat Enterprise Linux (RHEL) 5 , Kernel 2.6.18
    - http://www.redhat.com/rhel/server/mainframe/
- Debian
  - Debian Etch, Kernel 2.6.18
    - http://www.us.debian.org/ports/s390/
- Build your own
  - Download kernel from:
    - http://www.kernel.org

For more information about new hardware features available with latest kernel upgrades, see developerWorks navigation for Linux on System z at:

- http://oss.software.ibm.com/developerworks/opensource/linux390/index.shtml
- http://www-03.ibm.com/systems/z/os/linux/

3.3.1 31-bit and 64-bit options

Linux on System z solutions are available for 31-bit and 64-bit environments. The option availability depends on the System z model and the Linux distribution.

> **Note:** The choice between the 31-bit and 64-bit addressing environments can depend on the application or middleware you are planning to use. Some software does not support 64-bit addressing mode and may not work properly.
>
> The latest versions (SLES10 and RHEL5) of the distributions are 64-bits only distributions. Older versions (SLES9, RHEL4) are available as 31 or 64-bits distributions.

The Linux 2.6 kernel has undergone a substantial testing in 64-bit mode, and is intended for large-scale, highly available systems. The IBM strategy is focused on middleware applications running on 64-bit platforms. Whenever possible, a 64-bit enterprise Linux distribution is recommended. The 64-bit kernel offers greater memory addressability, and provides greater flexibility for running Linux on System z. With the 64-bit kernel, you can run many small Linux images or fewer but larger Linux images in a given z/VM partition. Applications that can benefit most from 64-bit addressing include:

- Databases
- Applications requiring access to large amounts of data
- Java applications

Although most middleware has been or will be ported to 64-bit, it is important to note that not all will benefit from the larger address space. In these cases, the middleware continues to operate in 31-bit addressing mode on a 64-bit Linux distribution using *compatibility mode.* For this reason there are 31-bit libraries available on 64-bit distributions as shown in Figure 3-1 on page 103.

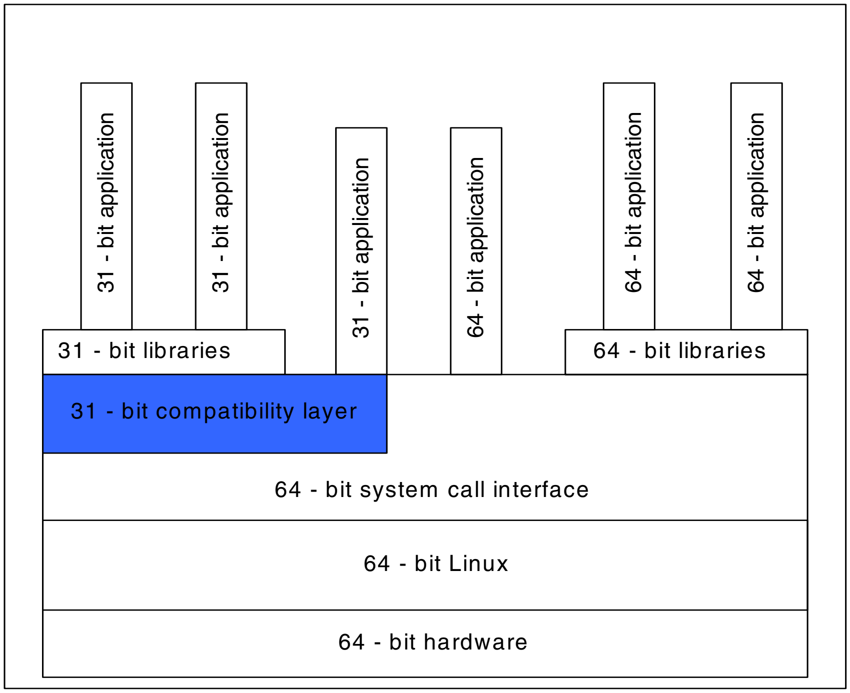

Figure 3-1 Compatibility mode system call emulation

3.3.2 Integrated Facility for Linux (IFL)

An IFL is a processor reserved for Linux (or Linux under VM). The significance is that it cannot be used to run other operating systems, and its existence is not reflected in the system model number, MIPS rating, or other capacity ratings. The system model, MIPS, or other capacity rating method has significant implications for software costs. Adding an IFL does not affect the costs associated with your system MIPS, which permits the use of Linux without impacting other software costs.

Note: IFLs must be used by LPARs running Linux or z/VM only. However they can be shared by multiple LPARs, provided that the LPARs are running Linux or z/VM.

IFL has the same functionality as a general purpose CP on the System z server. These inculde but not limited to: Support for On/Off Capacity on Demand (On/Off CoD), Capacity for Planned Event (CPE), and Capacity BackUp (CBU) for emergency situations.

Important: Figure 3-2 displays how IFLs may be allocated in a LPAR for pre-z10; however, z10 may have IFLs and General CPUs in the same LPAR. So in Figure 3-2 configuration A would be allowed in a z10 but not to pre-z10 models.

Figure 3-2shows how IFLs may be allocated to LPARs

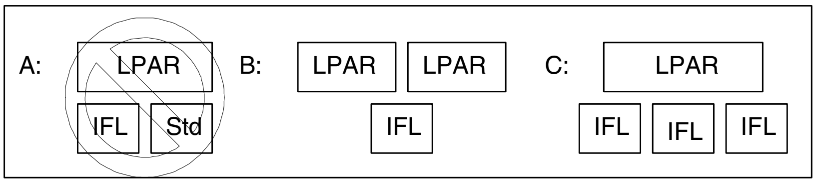

Figure 3-2 The relationship between IFLs and LPARs

For more details on IFLs please visist IBM's website at:

- http://www.ibm.com/jct03004c/systems/z/os/linux/ifl.html

3.4 Getting Started: The Basics of Linux

In this section we will go over some basic concepts on how to use your Linux server.

File system

Linux is made up of files and directories in a hierarchical structure. The Linux file system is divided up into key main directories which a user can navigate through.

- */home* - all users' home directories are stored here. This is basically where users store any file they own: media, source code, desktop configuration files and so on.
- */root* - Home directory for the root user
- */bin*, aka executables -- essential system programs are located here. Commands to load kernel modules and access devices are usually placed here.
- */usr/bin* - system commands (the real warehouse for software)
- */usr/sbin* - commands usually used to perform system administration tasks
- */usr/local* - Directory used as a second level of Linux FHS (Filesystem Hierarchy Standard). Software that follows the FHS but you don't want to mix with other software on the first level of the FHS are usually placed here. For example, an in-house software that you may be developing.
- */etc* - stores all of the configuration files for the system and programs' default configuration files that can be shared by all users. The runlevel configuration files are also placed here.
- */var* - logs, spool directories, database files, file servers files, system lock files.
- */lib* - Programs' libraries and shared libraries
- */dev* - device files: memory, disks, removable media devices (storage, multimedia) and any type of hardware at all.
- */proc* and */sys* - special system's filesystems. They contain realtime data about the configuration of the hardware and running processes as Linux sees them: processor, ACPI parameters, processes, information of each running process (PID, I/O file descriptors), kernel configuration
- */opt* - Location to store additional software. Usually, vendors applications are usually installed here.
- */tmp* - Folder for temporary files. Its contents are usually deleted during boot time. Never place important files here.

The root user

The root user is by default the system administrator, which means there are no restrictions on root privileges. Since root has such authority, someone who is not completely knowledgeable of the system can cause severe damage. It is only recommended to log onto root for administrative actions only. User root is equivalent to IBMUSER in z/OS but it is even more powerful because root has an access to everything by default. There is no way to restrict root access.

3.4.1 Basic starting steps

This section talks about basic commands, such as how to navigate around the directories, viewing the contents of a directory or file, and creating directories.

Note: It is important to note that Linux commands and file names are case-sensitive. For example, the command **chmod** can not be typed as **Chmod** or **CHMOD**. The same for filenames is true.

Hint: Pressing the *Tab* key after you start typing a command or file name can complete the rest of the command/filename. Also pressing *Ctrl-R* does a search of all past commands issued.

Getting help with Linux commands

In Linux there is no help menu such as there is in z/VM or z/OS; however, there is a command called **man** which displays the page of the command from the online reference manuals. This page includes: the name, synopsis on how it is used, description on how the command functions, and other details. To use the command, type **man commandname**, where **commandname** is the name of the command you want to learn about (See Example 3-1 on page 106). Use the arrows to scroll up and down the file and to quit simply press *q*.

Note: You must know the name of the command you want to use to access its manual page.

Example 3-1 The man command on the man command itself

```
lnxken:/ # man man
NAME
       man - an interface to the on-line reference manuals

SYNOPSIS
       man  [-c|-w|-tZHT  device]  [-adhu7V]  [-i|-I]  [-m
       system[,...]] [-L
```

```
         locale] [-p string] [-M path] [-P pager] [-r  prompt]  [-S
        list]  [-e
         extension] [[section] page ...] ...
         man -l [-7] [-tZHT device] [-p string] [-P pager] [-r prompt]
        file ...
         man -k [apropos options] regexp ...
         man -f [whatis options] page ...

DESCRIPTION
         man is the system's manual pager. Each page argument given to
         man  is normally  the name of a program, utility or function.
         The manual page associated with each of these arguments is then
         found and displayed. A section,  if provided, will direct man to
         look only in that section of the manual.  The default action is
         to search in all of  the  available sections,  following  a
         pre-defined  order and to show only the first page found, even
         if page exists in several sections.

         The table below shows the section numbers of the  manual
         followed  by the types of pages they contain.

         0   Header files (usually found in /usr/include)
         1   Executable programs or shell commands
         2   System calls (functions provided by the kernel)
         3   Library calls (functions within program libraries)
         4   Special files (usually found in /dev)
         5   File formats and conventions eg /etc/passwd
         6   Games
         7   Miscellaneous  (including  macro  packages and convenâ
             tions), e.g. man(7), groff(7)
         8   System administration commands (usually only for root)
         9   Kernel routines [Non standard]

         A manual page consists of several parts.

         They may be labelled  NAME,  SYNOPSIS,  DESCRIPTION,  OPTIONS,
         FILES, SEE ALSO, BUGS, and AUTHOR.
```

Navigating through the file system

The command to be able to change from one directory to another one is, **cd** (short for change directory). The proper use of the command is:

cd directoryname

where *directoryname* is the name of the directory you wish to access (See Example 3-2).

Example 3-2 The change directory command: **cd**

```
lnxken:~ # cd Desktop/
lnxken:~/Desktop #
```

When using the **cd** command you can also just type **cd ..** (with the space and two periods) which moves you up one level of the directory tree (see Example 3-3 on page 108). You can type **cd /** which brings you to the root of the file system directory no matter which directory you are currently in(see Example 3-4 on page 108). Finally you can type **cd -** which takes you back to the last directory you previously were in (See Example 3-5 on page 108).

Example 3-3 Changing to the parent directory: **cd ..**

```
lnxken:/etc/init.d/rc3.d # cd ..
lnxken:/etc/init.d #
```

Example 3-4 Changing to the root of the file sytem directory: **cd /**

```
lnxken:/etc/init.d/rc3.d # cd /
lnxken:/ #
```

Example 3-5 Changing to the previous directory: **cd -**

```
lnxken:/ # cd -
/etc/init.d/rc3.d
lnxken:/etc/init.d/rc3.d #
```

Displaying the contents of a directory

Now that you can change from one directory to the next, you will want the ability to look at what is inside the directories. To do this, use the **ls** command, which lists the contents in the current directory, or you can type **ls directoryname** (where directoryname is the name of the directory that you want to list the contents from) which list the contents of that directory (see Example 3-6 on page 109).When you use the **ls** command, just a list of file names is returned. To list the contents with long details you can use the command **ls -al** (see Example 3-7 on page 109).

> **Note:** The default rules for color of contents are:
>
> - directories = blue
> - files = white
> - executable files = green
> - links = light blue
> - devices = yellow
> - sockets = pink

Example 3-6 Listing the current directory contents

```
lnxken:/ # ls
bin   dev  home  lib64       media  opt   root  srv  tmp  var
boot  etc  lib   lost+found  mnt    proc  sbin  sys  usr
lnxken:/ # ls root/
.DCOPserver_lnxken_:3  .gconf           .local              .vnc
.DCOPserver_lnxken__3  .gconfd          .mozilla            .wapi
.ICEauthority          .gnome2          .qt                 Desktop
.Xauthority            .gnome2_private  .skel
.bash_history          .gnupg           .suse_register.log  bin
.exrc                  .kde             .viminfo
lnxken:/ #
```

Example 3-7 Listing all the current directory contents with details

```
lnxken:/ # ls -al
total 92
drwxr-xr-x 21 root root  4096 May 19 13:58 .
drwxr-xr-x 21 root root  4096 May 19 13:58 ..
drwxr-xr-x  2 root root  4096 May 15 09:50 bin
drwxr-xr-x  4 root root  4096 May 15 09:52 boot
drwxr-xr-x  9 root root  2440 May 19 11:57 dev
drwxr-xr-x 75 root root  8192 May 19 11:58 etc
drwxr-xr-x  3 root root  4096 May 15 13:54 home
drwxr-xr-x  9 root root  4096 May 15 13:26 lib
drwxr-xr-x  5 root root  4096 May 15 13:25 lib64
drwx------  2 root root 16384 May 15 09:43 lost+found
drwxr-xr-x  2 root root  4096 Jun 16  2006 media
drwxr-xr-x  2 root root  4096 Jun 16  2006 mnt
drwxr-xr-x  4 root root  4096 May 15 09:48 opt
```

```
dr-xr-xr-x 57 root root     0 May 19 11:57 proc
drwx------ 16 root root  4096 May 19 13:58 root
drwxr-xr-x  3 root root  8192 May 15 09:51 sbin
drwxr-xr-x  4 root root  4096 May 15 09:43 srv
drwxr-xr-x 10 root root     0 May 19 11:57 sys
drwxrwxrwt 10 root root  4096 May 19 15:15 tmp
drwxr-xr-x 13 root root  4096 May 15 09:50 usr
drwxr-xr-x 14 root root  4096 May 15 09:46 var
lnxken:/ #
```

Creating a directory

To create a directory, use the **mkdir** command which creates a directory in the current directory you are in. Type **mkdir directoryname** (see Example 3-8) to create the directory or **mkdir -mXXX directoryname** (see Example 3-9) to set the permissions for the directory, where XXX is the permission code of the directory (see Part 3.4.2, “File permissions” on page 115 for more details on permission codes).

Example 3-8 Making a directory

```
lnxken:~/Desktop # mkdir MyMusic
lnxken:~/Desktop # ls -al
total 48
drwx------  5 root root 4096 May 19 15:56 .
drwx------ 16 root root 4096 May 19 13:58 ..
-rw-r--r--  1 root root   69 May 15 13:48 .directory
-rw-r--r--  1 root root 1043 May 15 13:48 MozillaFirefox.desktop
drwxr-xr-x  2 root root 4096 May 19 15:56 MyMusic
-rw-r--r--  1 root root 1797 May 15 13:48 Network.desktop
-rw-r--r--  1 root root 2483 May 15 13:48 Printer.desktop
-rw-r--r--  1 root root  887 May 15 13:48 myComputer.desktop
-rw-r--r--  1 root root 5065 May 15 13:48 trash.desktop
lnxken:~/Desktop #
```

Example 3-9 Making a directory with certain file permissions

```
lnxken:~/Desktop # mkdir -m777 MyPictures
lnxken:~/Desktop # ls -al
total 52
drwx------  6 root root 4096 May 19 16:06 .
drwx------ 16 root root 4096 May 19 13:58 ..
-rw-r--r--  1 root root   69 May 15 13:48 .directory
-rw-r--r--  1 root root 1043 May 15 13:48 MozillaFirefox.desktop
drwxr-xr-x  2 root root 4096 May 19 15:56 MyMusic
drwxrwxrwx  2 root root 4096 May 19 16:06 MyPictures
```

```
-rw-r--r--  1 root root 1797 May 15 13:48 Network.desktop
-rw-r--r--  1 root root 2483 May 15 13:48 Printer.desktop
-rw-r--r--  1 root root  887 May 15 13:48 myComputer.desktop
-rw-r--r--  1 root root 5065 May 15 13:48 trash.desktop
lnxken:~/Desktop #
```

Moving files

To move files, the command is **mv source destination**, where source is the location and file name of the file to be moved and destination is the new location of the file (See Example 3-10).

Example 3-10 Moving a file from one location to another

```
lnxken:~/Desktop/MyDocs # ls -al
total 20
drwxr-xr-x 4 root root 4096 May 19 17:15 .
drwx------ 3 root root 4096 May 19 17:12 ..
drwxr-xr-x 2 root root 4096 May 19 16:29 MyMusic
drwxrwxrwx 2 root root 4096 May 19 17:13 MyPictures
-rw-r--r-- 1 root root   83 May 19 17:15 smilepic
lnxken:~/Desktop/MyDocs # mv smilepic MyPictures/
lnxken:~/Desktop/MyDocs # ls -al
total 16
drwxr-xr-x 4 root root 4096 May 19 17:16 .
drwx------ 3 root root 4096 May 19 17:12 ..
drwxr-xr-x 2 root root 4096 May 19 16:29 MyMusic
drwxrwxrwx 2 root root 4096 May 19 17:16 MyPictures
lnxken:~/Desktop/MyDocs # ls -al MyPictures/
total 12
drwxrwxrwx 2 root root 4096 May 19 17:16 .
drwxr-xr-x 4 root root 4096 May 19 17:16 ..
-rw-r--r-- 1 root root   83 May 19 17:15 smilepic
lnxken:~/Desktop/MyDocs #
```

Note: To rename a file, without changing its location, use the *mv* with the old and new filenames as arguments:

mv *oldname newname*

will rename the file oldname into newname, without changing its location.

Copying files

To copy files, the command is **cp source destination**, where source is the location and file name of the file to be copied, and destination is the new location and file name (See Example 3-11 and Example 3-12).

Example 3-11 Copying a file or make a backup

```
lnxken:~/Desktop/MyDocs/MyPictures # cp smilepic smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures # ls -al
total 16
drwxrwxrwx 2 root root 4096 May 19 17:18 .
drwxr-xr-x 4 root root 4096 May 19 17:16 ..
-rw-r--r-- 1 root root   83 May 19 17:15 smilepic
-rw-r--r-- 1 root root   83 May 19 17:18 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures #
```

Example 3-12 Copying a file from one location to another location

```
lnxken:~/Desktop/MyDocs/MyPictures # cp smilepic.bak
/root/Desktop/MyDocs/
lnxken:~/Desktop/MyDocs/MyPictures # ls -al ..
total 20
drwxr-xr-x 4 root root 4096 May 19 17:21 .
drwx------ 3 root root 4096 May 19 17:12 ..
drwxr-xr-x 2 root root 4096 May 19 16:29 MyMusic
drwxrwxrwx 2 root root 4096 May 19 17:20 MyPictures
-rw-r--r-- 1 root root   83 May 19 17:21 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures #
```

Deleting directories or files

To remove a file, the command is **rm filename**, where *filename* is the name of the file to be removed (See Example 3-13 on page 112). To remove a directory you need to type the command **rm -Rf filename**, where **-R** means to recursively delete everything in that directory and **-f** means force the action to happen (see Example 3-14 on page 113).

Example 3-13 Removing a file

```
lnxken:~/Desktop/MyDocs # rm smilepic.bak
lnxken:~/Desktop/MyDocs # ls -al
total 16
drwxr-xr-x 4 root root 4096 May 19 17:21 .
drwx------ 3 root root 4096 May 19 17:12 ..
drwxr-xr-x 2 root root 4096 May 19 16:29 MyMusic
drwxrwxrwx 2 root root 4096 May 19 17:20 MyPictures
```

```
lnxken:~/Desktop/MyDocs #
```

Example 3-14 Removing a directory

```
lnxken:~/Desktop/MyDocs # rm -Rf MyMusic/
lnxken:~/Desktop/MyDocs # ls -al
total 12
drwxr-xr-x 3 root root 4096 May 19 17:22 .
drwx------ 3 root root 4096 May 19 17:12 ..
drwxrwxrwx 2 root root 4096 May 19 17:20 MyPictures
lnxken:~/Desktop/MyDocs #
```

View a file's contents

To view the contents of a file, there are at least three options: **less**, **more**, and **cat**.

- **less filename**
  - Displays the file one page at a time with ability to scroll up, down, right, and left.
- **more filename**
  - Displays the file one page at a time and only allows you to scroll down through the file
- **cat filename**
  - Displays the whole file at once and goes back to the shell. The command **cat** can also be used to concatenate several files into one file.

Customizing commands

To customize command names, you can use the **alias** command. The alias command provides a way to make that "hard to remember" command easier to remember or to create short cuts for commands. Example 3-15 on page 113 demonstrates using the **alias** command to rename the **rm** command to something easier to remember - in this case, **erase**.

Example 3-15 The **alias** *command*

```
lnxken:~/Desktop/MyDocs/MyPictures # alias erase=rm
lnxken:~/Desktop/MyDocs/MyPictures # ls -l
total 8
-rw-rw-rw- 1 root root 83 May 19 17:20 smilepic
-rw-r--r-- 1 root root 83 May 19 17:18 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures # erase smilepic
lnxken:~/Desktop/MyDocs/MyPictures # ls -l
total 4
-rw-r--r-- 1 root root 83 May 19 17:18 smilepic.bak
```

```
lnxken:~/Desktop/MyDocs/MyPictures #
```

Summary

Table 3-1 on page 114 provides some basic commands to help you get started in Linux. For more commands and help with Linux, see the following helpful websites:

- http://www.unixguide.net/linux/linuxshortcuts.shtml
- http://blog.lxpages.com/ultimate_linux.html

Table 3-1 Summary of Basic commands

Command	Description	Example
`cd`	changes directory	`cd directoryname`
`ls`	list a directory contents	`ls directoryname`
`mv`	moves a file or directory	`mv file newlocation`
`cp`	copies a file or directory	`cp oldfile newfile`
`mkdir`	makes a new directory	`mkdire newdirectory`
`rm`	removes a file or directory	`rm filename`
`more`	displays a file content with limited navigation abilities	`more filename`
`less`	displays a file contents with full navigation abilities	`less filename`
`cat`	displays a file and can join files together	`cat filename`
`man`	gives help on a command	`man commandname`
`alias`	gives alias name to a command	`alias aliasname=command`
`pwd`	display working directory	`pwd`
`who`	list who currently logged on	`who`
`whoami`	report what user you are currently logged on as	`whoami`
`ps`	list all process currently running	`ps`

3.4.2 File permissions

In Linux, every directory and file has a set of permissions, which assigns users, groups and others the permission to read, write, and excute other users files based on authorization.

There are three ways of restricting file access in Linux:

- Read (**r**): the file/directory can only be read
- Write (**w**): the file/directory can have changes made
- Execute (**x**): the file/directory can be executed

For those three types of file access, there are three sets of users:

- User (**u**) - the person who is the owner of the file/directory
- Group (**g**) - the user/users who are members of the group which the file/directory is assigned
- Others (**o**) - all users that where not the owner or in the group of the file/directory

In the file permission file, the first character is the type of file which can be (see Table 3-2 on page 115):

Table 3-2 File types

Type of File	**Character**
regular file	-
directory	d
symbolic link	l
named pipe	p
socket	s
character device file	c
blocked device file	b

The file permission is displayed with the **ls -l** command and an explanation of the file permission's layout and positioning is in Figure 3-3 on page 116 and Figure 3-4 on page 116

Figure 3-3 File permission layout

Position	1	2	3	4	5	6	7	8	9	10
Layout	File	User			Group			Other		
Character	Type	Read	Write	Execute	Read	Write	Execute	Read	Write	Execute
Example	d	r	w	x	r	w	-	r	-	-

Figure 3-4 Layout of the **ls -l** *command*

Type & Permission	# of Links	File's Owner	File's Group	Size in Bytes	Last modification date	filename
-rwxrw-rw-	1	Ken	Student	652	May 20 10:05	smilepic

In Example 3-16 on page 116 the file *smilepic* has displayed **-rwxrw-rw-**, which means the owner has read/write/excute access, the group has read/write access, and others have read/write access.

Example 3-16 Sample of file permissions

```
lnxken:~/Desktop/MyDocs/MyPictures # ls -l
total 8
-rwxrw-rw- 1 ken student 83 May 19 17:20 smilepic
-rw-r--r-- 1 ken student 83 May 19 17:18 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures #
```

Note: That a user might have permission for a certain action for a file; however, they still might not have access to that action based on the permission of the directory the file is located.

Changing permissions

A user can change their own file permission; however, they can not change other users' file permissions. Note that the **root** userid does have the power to change all files' permissions. The Linux command to change a file permission is **chmod**.

The **chmod** command can be used in two ways: by using the octal system or with symbols.

Octal system

The octal system uses numbers to change the permission of the file. Once a user has a handle on the octal system, it is generally faster for the user than using symbols. Table 3-3 displays the combinations of numbers used to give certain permissions.

Table 3-3 Permission combinations

Value	Permission
1	---
2	--x
3	-wx
4	r--
5	r-x
6	rw-
7	rwx

Example 3-17 demonstrates the use of the octal system.

Example 3-17 Octal System mode for **chmod command**

```
lnxken:~/Desktop/MyDocs/MyPictures # ls -l
total 8
-rw-rw-r-- 1 root root 83 May 19 17:20 smilepic
-rw-r--r-- 1 root root 83 May 19 17:18 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures # chmod 750 smilepic
lnxken:~/Desktop/MyDocs/MyPictures # ls -l
total 8
-rwxr-x--- 1 root root 83 May 19 17:20 smilepic
-rw-r--r-- 1 root root 83 May 19 17:18 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures #
```

Note that we changed the file permissions from **-rw-rw-r--** to **-rwxr-x---** by using the command

```
chmod 750 smilepic
```

The number 7 in the first position indicates that we want to change the owner's access from read/write access to read/write/execute access. The number 5 in the second position indicates that we want to change the group's access from read/write to read/execute (no write access) and the number 0 in the third permission indicates a change from read access only for everyone else to no access at all.

Symbols

Using symbols with the **chmod** command takes four vaules:

- The users types it will effect
  - u for user, g for group, and o for others
- The type of change being made
  - - for remove permission, + for add permission, = for defining permission
- The type of access that will be applied
  - r for read, w for write, x for execute
- The file/directory that change will be applied

Example 3-18 demonstrates the **chmod** command using symbols.

Example 3-18 Symbol mode for the **chmod** *command*

```
lnxken:~/Desktop/MyDocs/MyPictures # ls -l
total 8
-rw-r-x--- 1 root root 83 May 19 17:20 smilepic
-rw-r--r-- 1 root root 83 May 19 17:18 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures # chmod g-x smilepic
lnxken:~/Desktop/MyDocs/MyPictures # ls -l
total 8
-rw-r----- 1 root root 83 May 19 17:20 smilepic
-rw-r--r-- 1 root root 83 May 19 17:18 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures # chmod go+rw smilepic
lnxken:~/Desktop/MyDocs/MyPictures # ls -l
total 8
-rw-rw-rw- 1 root root 83 May 19 17:20 smilepic
-rw-r--r-- 1 root root 83 May 19 17:18 smilepic.bak
lnxken:~/Desktop/MyDocs/MyPictures #
```

3.4.3 Job Control

Since Linux is a multitasking OS this means that a user can run multiple jobs or processes at one time. As stated earlier in this chapter, to see a list of jobs running, use the Linux command **jobs** or **jobs -p** which gives the process number of each job running. To see a list of all running processes you own, you can type **ps,** and to see all processes running, type **ps -e**.

Note: The words job and process can be used interchangeably.

Run a job in the background

Most commands do not take a lot of time; however, if a command will take some time to execute you can temporarily suspend the job to the background with the command

`CRTL - z`.

From here, you can execute other commands, but to get back to your original job you suspended into the background, simply type the command

`fg %jobname` (brings the process back to execution, but holds the shell) or

`bg %jobnumber` (sends the programs to the background).

End a job

If for some reason you wish to cancel a job, simple type

`kill jobnumber` or

`kill jobname`.

With kill you can also tell how you want to kill the process by typing

`kill -n jobnumber/jobname`,

where n is a number that is the signal type to send the process. Number 9 is to kill process. The number 9 is the highest level, meaning end the job immediately, while 15 means allow the job to clean up before terminating. Use the **`man`** command for more details on **`kill`**.

3.4.4 Linux editor: vi editor

This section introduces the vi editor. We describe the three operating modes of vi and how to switch from one mode to another.

Modes of operation

It is important to know that vi operates in three modes:

- **`Command mode`** You enter commands in this mode.
- **`Insert mode`** You insert characters at the cursor position.
- **`Replace mode`** You overwrite the character at the cursor position.

To switch from one mode to another:

- From Command mode to Insert mode, press one of these single characters:

`i, I, a, A, o, O`

Note: Typing a single character (of the listed set) immediately brings vi to Insert mode. Do not make the mistake of pressing Enter after that. If you do, it will be treated as input in the Insert mode.

- From Command mode to Replace mode, press R (again, without Enter).
- From Replace or Insert mode to Command mode, press Esc.

Figure 5-1 illustrates the three modes in which vi operates and the keys needed to go from one mode to the other.

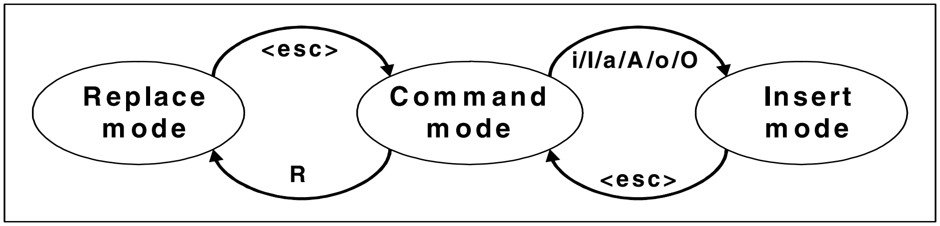

Figure 3-5 vi operating modes

Check the bottom left corner of the vi screen for an indication of the operating mode. There are three possibilities:

- `-- REPLACE --`
- `-- INSERT --`
- blank for Command mode

Normally vi starts in Command mode, and in the initial screen, the bottom left corner displays the filename until the first input is entered.

Some utilities like PuTTY (a free telnet and SSH client for Windows and Unix platforms which is found at http://www.putty.org) make vi easier to use. With PuTTY, for example, it is possible to switch to Insert mode from the other two modes. The Insert key will also bring you from Insert mode to Replace mode.

Commands for vi editor

This section contains a brief collection of the most frequently used vi commands.

Navigating around the editor

Listed in Table 3-4 on page 121 are different commands to move the cursor around the editior

Table 3-4 navigating around vi editor

Commands	Description
`0`	The beginning of the current line
`$`	The end of the current line
`Enter`	The beginning of the next line
`w`	The beginning of the next word
`G`	The first non-blank position of the last line

Insertion point

Remember that you can switch from command mode to insert mode. First, place the cursor and then type one of the following single-character commands; this establishes the insertion point as specified. After the command is entered the cursor appears at the insertion point. Table 3-5 on page 121 lists insert commands.

Table 3-5 vi insertion commands

Command	Description
`i`	Before the current character
`I`	Before the beginning of the current line
`a`	After the current character
`A`	After the end of the current line
`o`	At the beginning of a new line that will be inserted before the current line
`O`	At the beginning of a new line that will be inserted after the current line

Deleting

Listed in Table 3-6 on page 122 are different commands to delete

Table 3-6 vi deleting commands

Command	Description
`x`	The current character
`nx`	n consecutive characters starting from the current character
`dw`	The current word
`D`	From the current character to the end of the line
`dd`	The entire current line
`dtc`	The current line through the next occurrence of character *c*
`:id`	Line number *i*
`:i,jd`	Line number *i* until (and including) line number *j*
`:1,.d`	The first line of the file through (and including) the current line
`:.,$d`	The current line through (and including) the last line of the file

Note: In the last two commands the dot . refers to the current line. The "$" in the last command refers to the last line.

Moving and copying

Table 3-7 on page 122 lists the different commands that allow you to move and copy

Table 3-7 vi moving and copying commands

Commands	Description
`:i mj`	Move line *i*, insert after line *j*
`:i,jmk`	Move lines number *i* through *j* (inclusive), insert after line *k*
`:icoj`	Copy line number *i*, insert after line *j*
`:i,jcok`	Copy lines number *i* through *j* (inclusive), insert after line *k*

Replacing

Listed in Table 3-8 on page 123 are different commands to perform the replace function.

Table 3-8 vi replacing commands

Commands	Description
`rc`	Replace the current character by *c*
`:s/ppp/qqq/`	In the current line, replace the first occurrence of pattern *ppp* by *qqq*
`:s/ppp/qqq/g`	In the current line, replace all occurrences of pattern *ppp* by *qqq*

Important: In vi, words are delimited by space, but also by any character that is not a digit and not a letter.

Locating a string pattern

Listed in Table 3-9 on page 123 are different commands to locate a certain string pattern

Table 3-9 vi locating a string

Commands	Description
`/xxx <Enter>`	Find the pattern xxx
`/ <Enter>`	Find the pattern xxx of a previous /xxx command
`?xxx <Enter>`	Finds the pattern xxx (search direction is reversed towards the top of the file)
`? <Enter>`	Pattern xxx of a previous /xxx or ?xxx command

Note: It is possible to use a “regular” expression in the string pattern. For example [0-9] represents any digit and [a,e,i,o,u] represents any vowel.

Miscellaneous

Listed in Table 3-10 on page 124 are some miscellaneous commands

Table 3-10 vi miscellaneous commands

Commands	Description
j	Go to line j (cursor at the beginning of this line)
u	Undo the most recent change in the current line
U	Restore the current line to the state when it was first visited
xp	Transpose the current character with the one next to it
ddp	Exchange the current line with the next line
~	Toggle the case of the current character
n~	Same as above, but for n consecutive characters
:help ccc	Get information about command ccc

Note: The xp (transpose) command corrects a lot of typing errors by exchanging the order of two consecutive characters.

In case-sensitive programming languages, such as Java and C, many mistakes pertain to a wrong case. The ~ (tilde) command is convenient to correct this.

Saving and closing file

Listed in Table 3-11 on page 124 are different commands to save and close the file

Table 3-11 vi saving and closing commands

Commands	Description
:w	Write to file (save it)
:q	Quit the vi session
:!q	Quit without saving
:wq	Save first, then quit

3.5 Explore further

For more information about the vi editor, you can study the tutorial that is available on your system. Copy the file *vimtutor* from the /usr/bin directory (or from some other directory, depending on your installation) to your home directory. Then execute **vimtutor** from your home directory. In addition to the explanations, there are also exercises to practice the commands.

You can also refer to:

- http://atlas.scs.carleton.ca/~youngsoo/misc/vi.html

Here are additional useful links to additional information about Linux:

- z/VM and Linux for System z Resources
  - http://www.vm.ibm.com/linux/
- Linux for IBM System z
  - http://www.ibm.com/servers/eserver/zseries/os/linux/
- System z mainframe servers
  - http://www.ibm.com/servers/eserver/zseries/
- General Linux Information:

Linux Handbook A Guide to IBM Linux Solutions and Resources, SG24-7000

4

Installing z/VM and creating Linux or z/OS guests

In this chapter we introduce the concepts and steps required to perform z/VM installation, and then creating a Linux and z/OS guest to run under z/VM, from the moment you have finished using the HMC to create your LPAR to the point where you have guest systems up and running.

For information on how to use the HMC to wrap up the resources for your LPAR, please see *Hardware Management Console V7 Handbook*, SG24-7491 and *IBM System Storage DS8000 Series: Architecture and Implementation*, SG24-6786.

Objectives

On completion of this chapter, you should be able to:

- Install a z/VM Operating System onto your LPAR
- Understand the differences between z/OS and z/VM from an installation perspective
- Set up the environment in z/VM for a Linux virtual machine
- Install Linux in a z/VM virtual machine
- Create and run a guest z/OS system

4.1 Initial installation of z/VM - comparison to a z/OS installation

This section is intended for z/OS systems programmers with little or no experience with z/VM to help them understand the basics of installing a z/VM system.

4.1.1 Installation methods and deliveries

Both existing z/OS and z/VM sites would probably perform their installation or upgrade tasks from an existing running system.

z/VM sites would typically use an existing virtual machine to install a new z/VM system from a z/VM DDR (tape delivery) or DVD delivery. As of January 25, 2008, the z/VM base operating system and base options became available for Internet delivery for orders placed using ShopzSeries in countries where it is available. See http://www.vm.ibm.com/buy/edelivery/ for further details.

z/OS sites would do so from an existing z/OS image using one of the available installation methods offered by IBM, currently ServerPac, CBPDO or SystemPac®.

4.1.2 z/VM delivery

There is only one delivery type offered from IBM to install z/VM, although there is an option for type of media (DVD or tape).

4.1.3 z/OS deliveries

As mentioned, the available installation methods offered by IBM for z/OS are currently ServerPac, CBPDO or SystemPac®. Here, we offer a brief description of each.

ServerPac

The z/OS installation method entitled ServerPac is a "software delivery package consisting of products and service for which IBM has performed the SMP/E installation steps and some of the post-SMP/E installation steps" (according to z/OS V1R9.0 Planning for Installation). The package is an integrated set of products that have passed initial testing. This method involves the use of the CustomPac Installation Dialog. This dialog allows the customer to do some customization, for instance deciding the naming of data sets and volumes and the placing of data sets within the set of disks built during installation. Customers are given the choice of either installing an IPL-ready z/OS image (including the necessary auxiliary data sets; page, SMF etc) or just to install the products and establish these auxiliaries at a later time.

CBPDO

The Custom-Built Product Delivery Option (CBPDO) delivery consists of uninstalled and unintegrated service without any dialog or tools to assist the installation process. This has to be carried out by the customer by using SMP/E.

SystemPac

The last alternative, SystemPac, is a complete delivery of integrated products (even ISV products) to the customer's specification. It will contain an IPL-ready system, including all necessary auxiliary data sets. It comes in two formats; the full volume format or the dump-by-data-set format. This type of delivery would probably also be the only option for new z/OS customers.

Other deliveries

See your IBM service representative for more delivery types or packages offered by IBM. There are some that consist of a more customized nature, and may not be orderable from every region or country.

4.1.4 Comparing a z/VM installation to z/OS installation types

By comparison, the SystemPac full volume format type delivery would be very similar to a z/VM delivery in the event that a customer would install an operating system for the first time.

Both the SystemPac delivery and a z/VM DDR (tape) delivery (assuming a first time installation) would imply the use of stand alone (IPLing from tape) versions of the IBM Device Support Facility (ICKDSF) for formating disk volumes and DFSMSDSS for z/OS or DDR for z/VM to restore the volumes. These activities are performed from the Hardware Management Console (HMC). By comparison, installing z/VM from a DVD delivery appears to be the easiest method to someone already familiar with z/OS.

By using the z/VM DVD delivery you have the ability to bring up a **RAMDISK** operating system, loaded into the System z processor memory. This system is functionally equal to z/VM itself. This is invoked from the HMC using the Integrated 3270 Console facility. From within this operating system, you perform most of the remaining installation activities.

First you invoke the exec **INSTPLAN** to specify which products to install, which language to use and type and volsers of target DASD. Next you use the executable, **INSTDVD,** to perform the main installation of z/VM. Upon completion of this, you IPL from the newly installed disk and run the executable, **INSTVM,** to finish the installation.

This method offers both a comprehensive dialog and a stable installation environment.

Remaining activities after a z/VM installation

Usually, a z/VM delivery is accompanied by a Recommended Service Upgrade (RSU) tape or DVD. This should be installed after the initial installation has completed. It is installed from within z/VM using the executables **SERVICE** and **PUT2PROD**. For a discussion on these topics, refer to Chapter 9, “Applying system maintenance” on page 393.

Disk considerations

Note that if you have duplicate volumes, z/VM uses the volume residing on the lowest device number without giving you an option. This is opposed to z/OS, that gives you the option of varying a device offline.

> **Important:** It would be a good idea to avoid having duplicates and to establish a suitable standard for disk volume labels.

4.2 z/VM installation concepts and considerations

Before going through a z/VM installation you are presented here with some concepts that are used throughout. You should get familiar with these before reading 4.3, “Installing z/VM in an LPAR” on page 143. This will make you able to compare a z/VM installation and its required configuration to a typical z/OS installation and configuration.

4.2.1 Interacting with z/VM

Here we outline some of the differences between z/OS and z/VM in terms of interacting with the system.

A user familiar with z/OS would typically log onto the system and use an ISPF session to interact with the system[1]. This would be a menu-driven interaction with the system, in which they must navigate through the menu to get to a particular system functionality or command and execute it.

[1] For the purposes of this book, we make the assumption that the vast majority of z/OS users interact with it by the use of ISPF instead of native TSO.

z/VM users, on the other hand, have a different way of interacting with the system - through the Conversational Monitor System (CMS), which is a command-line like interface[2] that we describe below.

CMS

In a typical z/VM install, CMS is the code that gets IPLed after a user logs in. It was designed to facilitate mainframe virtual machine administration. From within it, you can issue commands to edit files, create, delete and manage them, execute programs and scripts. In a simplistic analogy, CMS allows you to perform the same basic set of tasks that you do at a Linux console. Figure 4-1 illustrates a 3270 session displaying the CMS as the user `MAINT` logs in. An overview of CMS is located in section 2.5, "Conversational Monitor System (CMS)" on page 39.

A CMS interface is depicted in Figure 4-1. Notice how similar CMS looks like to a Linux console session. Most users would be familiar with the concept of working with a console session, so don't think that interacting with CMS is much different than that.

```
LOGON MAINT
ENTER PASSWORD  (IT WILL NOT APPEAR WHEN TYPED):

z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0098 RDR,   NO PRT,   NO PUN
LOGON AT 14:35:40 EDT FRIDAY 05/09/08
z/VM V5.3.0    2008-05-06 13:40


                                                  VM READ    VMLINUX6
```

Figure 4-1 Login screen and CMS

You should be familiar with how to use the CMS help menu. Read section 2.5.1, "Using the help command in CMS" on page 40 for an introduction to the subject.

The following is a list of the commands you are most likely going to use to find files, copy and edit them.

- **The `FILELIST` command**

  The usual way to list files in z/VM is by calling the **`FILELIST`** command from within CMS. z/OS users would usually do this through a menu interface, ISPF.

[2] CMS is not a simple command shell, but an entire operating system by itself.

See, “The COPYFILE command” on page 79 for more details about this command.

- **The COPYFILE command**

  Use this command to create backup copies of the configuration files you will edit. You can find this command’s syntax in section “The COPYFILE command” on page 79.

- **The XEDIT command**

  In CMS, there are two ways available to edit a file:

  – when you already know the file name and where it is

    • This is the straightforward scenario. In order to edit file named `PROFILE EXEC` that you have on your A disk, you would type the following on CMS’s command line:

      `xedit profile exec a`.

  – when you do not know what the file name is or where it is, or both

    • In this scenario, you have to first search for the file you want to edit by using the **`FILELIST`** command so that you get a list of files. Place your cursor onto the line corresponding to the file you want to edit and press F11.

Although the three commands mentioned here are the ones you will need to manipulate files throughout the installation, it is recommneded that you spend a few more minutes having a look at section 2.8.2, “CMS” on page 76 to learn about other commands. Also, reading the appropriate chapter of the Redbook *Introduction to the New Mainframe: z/VM Basics*, SG24-7316 will help you get familiarized with z/VM commands.

4.2.2 System files

As within every system, z/VM has configuration files. They are used to define minidisks for users, to define virtual resources such as virtual switches, to define profiles for `CMS` and XEDIT sessions and so on. In this section you will learn about the following files:

- USER DIRECT
- SYSTEM CONFIG
- PROFILE EXEC
- PROFILE XEDIT

You will learn how to access and modify each of these files when the time comes throughout the installation.

z/OS analogy: The configuration files discussed here are the z/VM counterparts of z/OS `SYS1.PARMLIB` concatenation, `RACF` definitions of users and your initial `TSO/ISPF` routine invoked from your logon procedure JCL

The USER DIRECT file

This is the z/VM configuration file that handles the definitions for users and their minidisks, DASD space allocation, and definitions for common profiles that users may import into their user settings. Such profiles are meant, among other things, to create basic devices such as the *reader* and *punch* spoolers needed by every user and to create links to other users disks, in particular to some of `MAINT`'s disks, in order to access critical code read-only, such as the CMS, without duplication into their mini disks. This file is located on MAINT's 2CC disk.

The SYSTEM CONFIG file

This file holds most of the configuration for the system. It is in this file that you set up which DASD volumes will be owned by the system, for example the ones that are dedicated to paging, and which ones are user owned, for example volumes used to host Linux virtual machines. It is also used to set up virtual resources, such as virtual switches used to serve networking to Linux VMs, and to limit the amount of resources given to ordinary users, such as the number of minidisks each may own. This file is located on MAINT's CF1 disk.

The PROFILE EXEC file

A `PROFILE EXEC` is different from other executables (also known as execs). It has the special file name, `PROFILE`, and it is automatically processed whenever you enter `IPL CMS` (or if you have an automatic IPL mechanism in place).

Your `PROFILE EXEC` contains the CP and CMS commands that you enter at the start of every terminal session. It can be used to set up special characteristics for each CMS user beyond those defined in that users directory entry, such as:

- Making any non-standard minidisks or shared file systems in your virtual machine configuration known to CMS
- Changing the colors used to display data at a terminal (color terminals only)
- Making frequently used execs storage-resident
- Setting up to 24 program function keys for commonly used commands
- Changing the Ready message sent by CMS
- Automatically checking your virtual card reader (your “in-tray”) for files and messages
- Taking you directly to an application instance

Each z/VM user has its own PROFILE EXEC file. This is similar to the Linux `.bashrc` file, and the z/OS initial logon routine (REXX or CLIST scripts). Every time a user logs in, this file is accessed and any special configurations and commands in it will be set or run. It is a good practice to maintain the same profile among users since this will ensure a homogeneous look and feel when you log in to different user accounts. This file is located on the 191 disk of each z/VM user.

The `PROFILE XEDIT` file

This is the configuration file for the `XEDIT` editor and is accessed everytime it is invoked by the user. As you have seen in , “Customizing XEDIT” on page 95, a good customization of this file will be critical for a better experience using XEDIT. This file is similar to the `.vimrc` file in Linux for its `VI editor`. In z/OS this is similar to profile settings set up in the `ISPCFIGU` module. This file is located on the 191 disk of every z/VM user.

4.2.3 Important z/VM installation userids

As with every system, there are some user accounts that have a special meaning. Here we outline some of them, the ones used throughout a z/VM installation, and make a comparison to z/OS and Linux.

- The MAINT user
- The AUTOLOG1 user
- The TCPIP user

The MAINT user

This is z/VM’s super user. It is analogous to the a z/OS user with access to system proviledged information through RACF, and also analogous the Linux `root` user.

This z/VM user has access to all of the priviledge classes mentioned in “z/VM privilege classes” on page 287. It is granted, therefore, access to all of the system’s configuration files and management commands, such as granting a user access to a virtual switch.

The AUTOLOG1 user

This is a special user in z/VM and there is no comparable user in either z/OS or Linux. The closest concept to it in Linux are the runlevels, which are responsible for bringing up services during startup.

Running commands or scripts automatically right after z/VM IPLs is accomplished through the `AUTOLOG1` z/VM user. As mentioned in 2.2.1, “History of z/VM” on page 18, the concept of a z/VM user maps to a virtual machine. So, the `AUTOLOG1` *virtual machine* is responsible for starting other virtual machines up (logging other users in) as soon as the z/VM IPL finishes. This is usually done to bring the TCP/IP virtual machine up, for example.

Note: If the **`noautolog`** parameter is *not* specified when z/VM is IPLed, the `AUTOLOG1` virtual machine is started. Because this virtual machine IPLs CMS, the `PROFILE EXEC` that is found on its A disk is run. This is analogous to the `/etc/profile` file on Linux and the `autoexec.bat` on DOS systems.

z/OS analogy: The tasks performed by the z/VM AUTOLOG1 user are similar to using START statements in members IEACMDxx and COMMNDxx in the SYS1.PARMLIB concatenation

The TCPIP user

In z/OS, the TCP/IP stack is managed by the z/OS Communications Server in a separate address space. In z/VM, however, the stack is managed by a virtual machine: the `TCPIP` virtual machine (defined as the `TCPIP` user).

4.2.4 Dealing with disks

A file hierarchy is run in CMS when a z/VM user logs into the system. It is a different hierarch than than used by Linux or z/OS. In fact, there is no hierarchy in CMS, but the concepts of file name, file type and file mode. Every file name is formed by these three parameters. You can think of this file system as a flat file system (or sequential) where you have only one level.

The first parameter, the file name, should be familiar to most users because it is just the name of the file. In CMS, file names are limited to 8 characters in length and are case insensitive.

The second one, file type, is analogous to the file extension of systems like Linux. In CMS, these are also limited to 8 characters in length and are case insensitive. In an analogy, the file name and file type together form something similar to the z/OS **filename.extension** (with a space instead of a dot between them) file naming convention of other systems.

The third concept, the file mode, is the parameter to specify where the file is located, that is, on which disk mapping it resides on. The file mode is one character long and is limited to the characters A-Z. The file mode is what makes the whole filesystem a flat universe because you are limited to having 26 simultaneous “places” to place your files.

We mentioned the word *mapping* in the previous paragraph, but before we talk about mapping a disk to a file mode, we need to discuss how disks are organized. Disks are nothing more than storage space. They hold files, and the files on them are identified by their name and type. However, as opposed to systems like Linux, disks are not identified by device names, they are identified by device IDs. Some of these IDs are well known IDs, such as `MAINT`'s 193 disk, which contains useful programs like **`CPSYNTAX.`**

The disk ID is an analogy to a template directory name in systems like Windows. For example, two z/VM users can have 191-labelled disks. Think of this as the *`My Documents`* folders that two different users may have in Windows. Both folders have the same name, *`My Documents`*, but each of them belongs to a different user and each of them denotes a different area in disk storage. The only difference here is that you are using numbers in CMS as opposed to names to denote a permanent storage area, that is, a disk.

Having said that, there are some interesting aspects of disks that can be further explored. If, for example, you want to access the **`CPSYNTAX`** command and you are not logged in as `MAINT`, you can *link* `MAINT`'s 193 disk into your user space so that you can access that command. The way you do this is by using the **`LINK`** and **`ACCESS`** (or its abbreviation: **`acc`**) commands.

Access another user's disks (linking)

The sequence of commands to link another user's disks to your user space is shown in Example 4-1.

Example 4-1 Linking and accessing disks

```
link autolog1 191 1191 rr
Ready; T=0.01/0.01 16:02:04
acc 1191 f
DMSACP723I F (1191) R/O
Ready; T=0.01/0.01 16:02:08
```

The **`LINK`** command shown in the example links disk 191 of user `AUTOLOG1` to our user space under the label *1191* in read only mode. This means that we can actually access the files on `AUTOLOG1`'s 191 disk by referencing our own *linked* 1191 now. However, to perform that we still need to *map* our 1191 disk to a *file mode*, so that we can use a triplet to access the files on it. Mapping a disk to a file mode is what we performed with the **`ACCESS`** command[3]. We mapped our 1191 disk to one unused file mode that we had available, which in this case was *F*. Figure Figure 4-2 shows what we have just performed here.

[3] In the z/VM world, mapping a disk as we defined the term is actually known as "accessing" a disk. We used the word map here to make the analogy easier to be understood by people not familiar with z/VM. From this point on, we will refer to disk mapping as accessing disks.

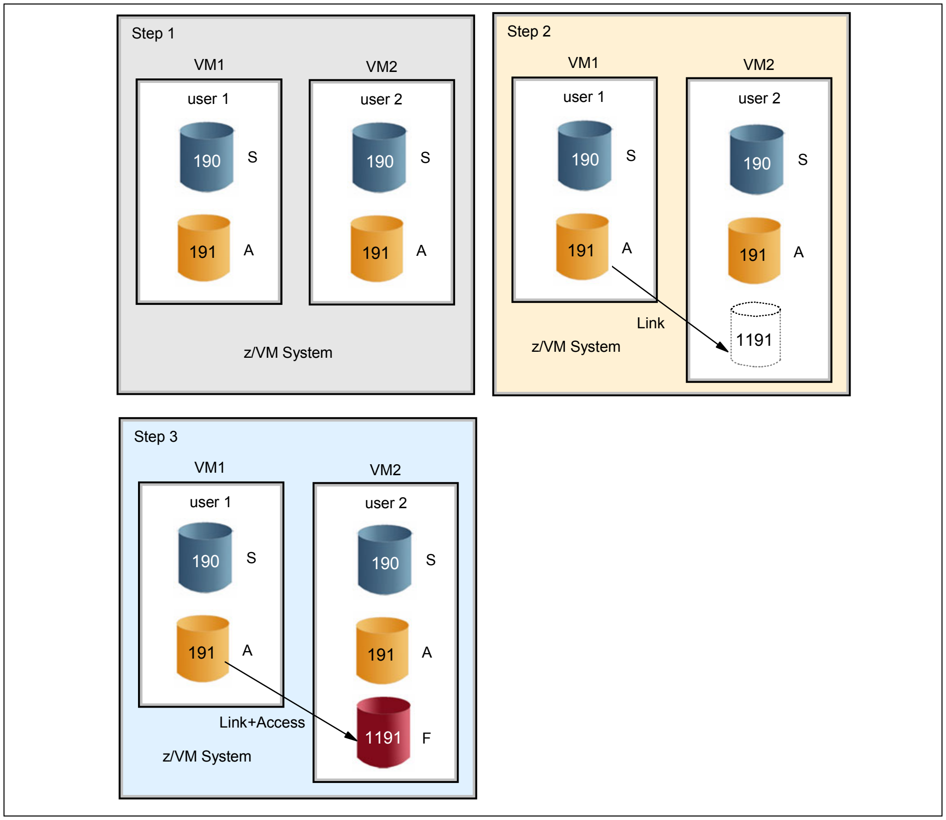

Figure 4-2 Accessing other users' disks.

To check that `AUTOLOG1`'s 191 disk is linked and mapped to our user space, we can run **`QUERY`**[4] **`DISK`** and check if there is an entry for our newly mapped *f* file mode. The output of the command is depicted in Figure 4-3 on page 138. Notice that the *f* file mode is our virtual disk with ID 1191 and that its label is `AUT191`, as we would expect.

Note: If you try to link to another user's disk and you are not logged in as `MAINT`, you may need to enter a password for the access mode (read, write) you are linking the disk in. Refer to "`Creating a disk`" `on page 140` for more detail on access modes.

[4] The abbreviation for the **`QUERY`** command is **`Q`**.

```
LOGON MAINT
z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES: 0113 RDR,   NO PRT,   NO PUN
LOGON AT 09:31:14 EDT TUESDAY 05/13/08
z/VM V5.3.0    2008-05-06 13:40
link autolog1 191 1191 rr
Ready; T=0.01/0.01 09:32:11
Ready; T=0.01/0.01 09:32:11
acc 1191 f
DMSACP723I F (1191) R/O
Ready; T=0.01/0.01 09:32:20
query disk
LABEL VDEV M STAT CYL TYPE BLKSZ FILES BLKS USED-(%) BLKS LEFT BLK
TOTAL
MNT191 191  A R/W 175 3390 4096 10 28-01 31472 31500
MNT5E5 5E5  B R/W 9 3390 4096 131 1288-80 332 1620
MNT2CC 2CC  C R/W 5 3390 4096 54 434-48 466 900
MNT51D 51D  D R/W 26 3390 4096 273 1245-27 3435 4680
AUT191 1191 F R/O 1 3390 4096 2 9-05 171 180
MNT190 190  S R/O 100 3390 4096 687 14592-81 3408 18000
MNT19E 19E  Y/S R/O 250 3390 4096 1010 26738-59 18262 45000
Ready; T=0.01/0.01 09:32:23


                                                    RUNNING   VMLINUX6
```

Figure 4-3 Checking that disks have been linked and mapped

AIX-like systems users could think of linking and accessing as soft linking to another user's directory. This naturally brings the idea of facing a file mode as a mapped directory to our flat CMS file system. For those readers who thought that living without directories would be a hard thing in z/VM, you see that you can manage 26 different directories at the same time. Just keep in mind that each of these *directories* are actually disks.

To *release* a linked disks from your user space, use the **RELEASE** command (or its abbreviated form: **REL**). To release the *f* disk that we just accessed, the command is **RELEASE f**.

Disks conventions

Now that you have learned the basic concepts surrounding disk access, this section describes some of the conventions related to disk access.

A CMS user will usually have at least two disks (probably minidisks, which you will learn about in a few moments) available. Typically with CMS, a user has a pair of virtual disk with addresses 191 and 190. Note the following points:

The 191 disk is conventionally accessed with mode A.

The 190 disk is accessed as S. The 190 minidisk is sometimes known as the “CMS system residence volume”, because it is where the CMS nucleus resides on disk.

Your A disk

CMS assumes that you have disks at addresses *190*, *191*, and *19E*. It also looks to see if you have a disk at address *192*. When CMS loads, these disks are accessed as follows:

- 191 becomes the A disk
- 192 becomes the D disk (if it exists)
- 190 becomes the S disk
- 19E becomes the Y/S disk

This will only succeed if your user has DASD with a virtual address of 191. The *A* disk is a work disk for a user’s permanent file storage. Generally the A disk of CMS users is the only writable disk they have access to. This would be similar to the home directory of users in systems that account user DASD space.

The A disk is exceptionally important because when CMS is loaded, it looks for a file named `PROFILE EXEC` on your A disk and attempts to execute that file.

The `PROFILE EXEC` is often modified to customize the CMS instance for a given user, or even to load another operating system like Linux.

Running out of space

Each virtual disk is divided into blocks that are usually 4096 bytes in size (although some disks, such as the help disk, use smaller blocks to save wasted disk space from the number of small files on them).

The minimum size of each file is one block. The maximum is limited by the size of the minidisk. Each user normally has read access to a number of these types of disks. But often, users run into trouble when they have no space left to write their own files.

For this reason, it is important to know how much space you have left on one of your accessed disks. You can get this and other useful information by using the **Query DISK** command.

> **Note: QUERY DISK** is different from the **QUERY DASD** command. QUERY DASD is a CP command for *devices*. Query DISK is a CMS-only command used for viewing your *accessed volume*s.

It is possible to completely fill up a disk, such as your A disk. In this situation, the system does not give you more space automatically. You must take some action to get more space or erase unnecessary files.

To acquire more space you could attach a new DASD volume, and then format and access it. Another option could be to link to another disk you have write access to or to create a temporary disk if you don't need to persistently save the data. More information on this is provided in 5.3.9, "Managing DASD" on page 259.

Creating a disk

In this section you will learn how to create a disk and what the concepts behind are. Later, you will lear how to format a disk.

The MAINT user in z/VM has access to many interesting system configuration files. Some of them are used for letting the system know which 3390 DASDs to automatically bring online in system boot, some are common profiles for ordinary users, and others are for managing the creation of mini-disks in those DASDs. In this section we are interested in creating mini disks for our users. A mini disk is a portion of the 3390 disk.

The file that assigns mini disks to users is called USER DIRECT and it can be found on MAINT's *2CC* disk, which is accessed using the *C* file mode. So, let's start by analizing the portion of this file in which we are interested. Figure 4-4 shows the mini disks for user CERON[5].

[5] We used an ordinary user as an example instead of MAINT because the are too many disks definitions and links for MAINT, which could lead to confusion.

```
 USER DIRECT   C1  F 80  Trunc=72 Size=2041 Line=1967 Col=1 Alt=0

01967
*********************************************************************
01968 USER CERON  CERON   32M  64M G
01969  INCLUDE IBMDFLT
01970  ACCOUNT ACT4 CMSTST
01971  MACH XA
01972   IPL 190
01973  MDISK 191 3390 2040 010 LX6W02  MR READ    WRITE   MULTIPLE
01974 *
01975
*********************************************************************
====>
```

Figure 4-4 Disk definitions for an ordinary user

The first line, outlined as line number 1968 in the file, defines the user `CERON` and sets his password to the word *CERON* as well. It is not a good practice to create such simple passwords, but for the sake of the examples throughout this book we will set passwords to be the same as the user name. The following two parameters, *32M* and *64M* indicate the initial and maximum amount of system storage (main memory) that this user's virtual machine will use. The last *G* in that line means the class privileges that this user will have access to, which in this case is the smallest possible. More about creating a user will be mentioned in , "Create the Linux Maintenance user" on page 188.

The second line, outlined as *1969*, includes a default profile called `IBMDFL` to the user definition. If you later have a look at the beginning of the `USER DIRECT file`, you will notice that this profile links some of `MAINT`'s disks to this user's user space. For example, it links `MAINT`'s 190 disk to its own 190 disk in read-only mode. This will allow this user to IPL CMS when he logs in.

On line *1973* you can actually see the definition of the mini disk for that user. It defines its 191 disk, on the 3390-3 type DASD labelled `LX6W02` starting at cylinder 2040 and being 10 cylinders in size in multi-read mode. Additionally, it defines three passwords, `READ`, `WRITE` and `MULTIPLE` for user who want to link to that disks in read, write and read-write modes respectively.

Once a disk definition is inserted for a given user, changes will not take place until you tell the system to perform them. This is actually accomplished by the **`DIRECTXA USER`** command, but it is always a good practice to check out our mini disks layout for gaps and overlaps before you actually commit your changes[6]. To check your changes, run **`DISKMAP USER`**, which will create a file named `USER`

DISKMAP in your *A* file mode. Check it out and look for the new user mini disk definition. Figure 4-5 shows the lines in that file referent to our newly created user.

```
 USER   DISKMAP  A1  F 80  Trunc=80 Size=382 Line=231 Col=1 Alt=0

00231
00232 VOLUME USERID CUU DEVTYPE START END SIZE
00233 LX6W02 $ALLOC$ A03 3390 00000 00000 00001
00234 40SASF40 2D2 3390 00001 00150 00150
00235 OSADMIN2 191 3390 00151 00160 00010
00236 OSADMIN3 191 3390 00161 00170 00010
<...snap...>
00318 COSTA 191 3390 01806 01815 00010
00319 HAIMO 191 3390 01816 01825 00010
00320 1826     1999     174   GAP
00321 GUILL 191 3390 02000 02009 00010
00322 OMAR 191 3390 02010 02019 00010
00323 RAY 191 3390 02020 02029 00010
00324 KEN 191 3390 02030 02039 00010
00325 CERON 191 3390 02040 02049 00010
====>
```

Figure 4-5 Checking DASD layout for gaps and overlaps

Notice in Figure 4-5 that the first line in bold shows a gap starting at cylinder 1826 and running up to cylinder 1999, for a total of 174 cylinders. Although you should try to avoid gaps to prevent fragmentation, they are harmless. The second bold line shows our newly created user, and there's no overlap between it and any of its surrounding mini disks definitions. Having checked that, we must run **DIRECTXA USER** for the changes to take place.

Formatting a mini disk

After a new user is created, its owner should format its mini disk space assigned to it so that files can actually be created onto that disk. This is accomplished by running the **FORMAT** command after the user logs in:

```
format 191 a
```

The previous command will format the user's 191 disk and use it as the *A* file mode, as shown in

[6] Gaps will only leave empty disk space between minidisks, but an overlap occurs when one minidisk definition overlaps into following mini disks, and z/VM actually allows you to do so.

```
LOGON GUILL
z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES:   NO RDR,   NO PRT,   NO PUN
LOGON AT 12:28:57 EDT TUESDAY 05/13/08
z/VM V5.3.0   2008-05-06 13:40

format 191 a
DMSFOR603R FORMAT will erase all files on disk A(191). Do you wish
to continue?
Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
gui191
DMSFOR733I Formatting disk A
DMSFOR732I 10 cylinders formatted on A(191)
Ready; T=0.01/0.01 13:14:27

                                                RUNNING   VMLINUX6
```

Figure 4-6 Formatting a mini disk

Note: When you log on to your user account for the very first time, your 191 disk (the A disk) will probably not have been formatted. You can check whether you have a suitably formatted A disk by executing the command **FILELIST**[a]. If you see a file listing after executing the **FILELLIST** command, then it has already been formatted.

However, if you receive the error message `HCPCMD001E Unknown CP command: FILELIST` then you may need to format your 191 disk before continuing. In that case, use the command **FORMAT 191 A** to place a VM readable file system on your A disk.

a. The abbreviated form for the FILELIST command is **FILEL**.

4.3 Installing z/VM in an LPAR

This section discusses installation of z/VM in an LPAR. For more detailed installation instructions, please see

http://www.vm.ibm.com/install/vm53inst.pdf

4.3.1 Planning

Before you start your z/VM and Linux installations you will need to plan to make sure you have everything you need in terms of hardware and software. In summary, you will need the following resources:

- Hardware
- Software
- Networking

Hardware resources

The following hardware is needed:

- A System z logical partition (LPAR); z800, z900, z890 or z990, System z9® or System z10
  - Processors or CPUs: One IFL (or CP) minimum, two or more are strongly recommended. In the z9 and below systems you are not able to mix general CPUs and IFLs.
  - Memory: 3 GB central/1 GB expanded minimum, 6 GB/2 GB or more recommended. This 3:1 ratio of central to expanded storage is a good starting point for all but the very largest systems.

    See the following Web site for a discussion of how to apportion memory:

    `http://www.vm.ibm.com/perf/tips/storconf.html`
  - DASD: 12 3390-3s or 5 3390-9s at a minimum
  - Open Systems Adapter (OSA) network cards: One card minimum with 8 device numbers (technically 6, but OSA "triplets" usually start on an even address). Two OSA Express cards with eight device numbers on one and four on the other is recommended for high availability.
- A network-attached computer that will act as a file server server of Linux distributions with at least 3GB of disk space plus 6GB for each Linux distribution that you intend to serve, but more may be needed. Setting up a server is *not* described in this book, however we reference a good literature in the following section, "Software resources".
- A workstation or desktop that has network access to the mainframe.

Software resources

The following software resources are needed:

- z/VM 5.3 install media with documentation (installation from DVD is described in this book).
- Linux install media (SLES or RHEL)

- An operating system for the file server
  - You can easily set up a file server using an existing Linux environment. The *z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES9*, SG24-6695 redbook has instructions on how to set up an NFS one.
  - Additional options to an NFS server are HTTP or FTP, served via the Apache server or via usual FTP servers found on Linux distributions. Instructions on how to set Apache up can be found at http://www.apache.org/
- The code associated with this book
- Tools on the workstation and desktop:
  - A 3270 Emulator such as *Attachmate Extra*, Hummingbird *Host Explorer*, or IBM *Personal Communications* for Windows desktops (for Linux desktops, a 3270 emulator named *x3270* is available)
  - A Linux SSH client such as PuTTY (recommended) or TeraTerm (for Linux desktops the **ssh** client is built in)
- A VNC viewer

Networking resources

The following network resources are needed:

- A TCP/IP address for z/VM
- One TCP/IP address for each z/OS, z/VM or Linux guest operating system
- Associated TCP/IP information:
  - DNS host name
  - DNS domain
  - DNS server TCP/IP address
  - TCP/IP gateway
  - TCP/IP subnet mask
  - TCP/IP broadcast address (usually calculated from address and subnet mask)
  - TCP/IP MTU size

The TCP/IP addresses should be routed to the OSA card(s).

z/VM users and password planning

Good passwords are critical to good security. However, requiring many different passwords generally leads to people writing them down, which clearly detracts from good security. Sometimes it is difficult to balance these two extremes.

This book considers different system administration roles:

- The z/VM system administrator
- The Linux system administrator
- The Linux virtual server end users

The z/VM and Linux system administrator may be the same person.

Note: For the purposes of this book, all passwords have been set to be the same as of the user name but it is important to realize that in your company environment, it is a risky thing to do.

You may want to define a finer granularity for passwords based on the following system administration roles:

- The main z/VM system administrator (`MAINT`)
- The z/VM network administrator (`TCPMAINT`)
- The z/VM Linux administrator (`LNXMAINT`, Linux guests administrator)
- The Linux virtual server end users (with or without access to 3270 sessions, with or without the root passwords)

The sets of passwords that you define will depend on the roles that your organization will adopt.

Note: If you wish to move your existing z/OS system onto z/VM and run it as a z/VM guest system (see 4.5, “Running a z/OS image as a z/VM guest” on page 217), you also need to define an administrator password for the z/OS guest virtual machine.

Disk planning

There are different aspects to consider when planning how to choose and allocate disk storage. Some aspects include the following

- Conventional ECKD™ DASD vs. FBA disks over SCSI/FCP
- 3390-3s vs. 3390-9s or large disk support
- Amount of disk storage per Linux image and how to allocate file systems

DASD vs SCSI/FCP

This book describes how to use conventional ECKD DASD and does not discuss FBA disks accessed over SCSI/FCP. This is not because either technology is superior, but simply because DASD seems to be much more common than SCSI/FCP disks. Sometimes a combination of these two types of disk storage is used - when that is the case the ECKD emulated DASD is often used for the root file system and SCSI/FCP disks are used for large data storage areas.

3390-3s vs. 3390-9s

Emulated 3390-3s format to about 2.3GB, while 3390-9s are three times the size or about 6.8GB. Either size will work, though 3390-3s have been recommended over 3390-9s by some performance analysts. This book mainly describes using 3390-3s, however, comments are added where using 3390-9s differs - especially when installing z/VM.

Disk storage per Linux image

In order to be able to install a graphical environment (the K Desktop Environment) into our Linux guest and also to leave it with lots of free space for general purpose tasks that the authors would like to play around with, we decided to use about 10GB of storage for the guest. The amount of space you will need for your particular goals with Linux on z/VM may vary, but having a root partition (/) around 3GB to 4GB at least is recommended for any Linux installation because, with the extra space left, you may easily grow later from a minimal package set to a full package set installation.

In the examples in this book, five 3390-3, each comprising 3339 cylinders, are allocated for the Linux image. Often recommendations are made to create many file systems to be mounted over directories other than the root file system itself (/), such as `/usr/`, `/var/`, `/tmp/`, `/home/`, `/opt/`, etc. We will not do this for all of them in the examples because:

- We are not optimizing the installation for any particular use. For more information about optimizing Linux partitioning, read *Linux Performance and Tuning Guidelines*, REDP-4285.
- We initially have only 10GB available for the installation, and segregating storage with that little space can lead us to too many small partitions that will bring the system to a useless state very quickly.

One common argument for having many mounted file systems is so that the root file system will not fill up and bring the system to a halt. For example, in a production environment where the Linux guest is acting as a file server, remote attacks may cause the logs in /var to consume all of the space ofthe partition it resides on. This may halt the system if /var resides on the same partition as the root partition (/).

> **Note:** Having a good partitioning plan is the first step to having an optimized and reliable Linux guest.

Memory planning

Planning memory may be the most difficult issue with Linux on System z and z/VM, yet the most important to ensure adequate performance. The simplest solution may appear to involve having enough central memory (storage) in the

LPAR so that z/VM never pages and Linux never swaps. However, such resource is often not realistically available. A good rule of thumb is to allocate memory on a just enough basis for each Linux server. A good starting point is to set a virtual machine size by changing the memory allocation value at just over the value at which the guest starts to swap at the Linux system level when under normal loading. If some level of sustained swapping is inevitable due to the nature of the workloads, then ensure virtual disks are used for the swap media.

An understanding of memory planning is definitely recommended, here are some books that cover this very important topic:

- The Redbook *Linux on IBM System z: Performance Measurement and Tuning*, SG24-6926 on the Web at:

  http://www.redbooks.ibm.com/redpieces/abstracts/sg246926.html?Open

- The IBM z/VM Performance Resource pages in general, on the Web at:

  http://www.vm.ibm.com/perf/

- The IBM z/VM page specifically discussing memory allocation:

  http://www.vm.ibm.com/perf/tips/storconf.html

One rule that can be recommended is to only have as few virtual machines logged on (or disconnected) as possible to handle the workload being presented. Every virtual machine that is not required should be be logged off where appropriate, as this will mean more memory for the other virtual servers which remain running.

We will set up our examples with Linux guests that are allocated 512MB of initial storage plus more 1.5GB of extended storage (for Linux swap). This is more than enough for the purposes of this book.

Planning worksheets

This section presents a worksheet with the values of the resources used throughout the examples in this chapter. You can use this as a reference to fill up your own planning worksheet on Table B-1 on page 409.

Note: All of the examples throughout this book that run commands which require resource information will encapsulate the resource value within the <> characters. When you are running the commands in your system, make sure to replace them with the values for your resources.

Table 4-1 z/VM resources worksheet

Name	Value	Comment
LPAR name	VMLINUX6	Storage: 4GB + 2GB Ext Processors: 4 DASD: 11 3390-3 disks
CPC name	SCZP201	Name of CPC on which the LPAR is located
z/VM system name	VMLINUX6	Name to be assigned to z/VM system
TCP/IP hostname	vmlinux6	Assigned by a network administrator; helpful to set in DNS beforehand, but not necessary
TCP/IP domain name	itso.ibm.com	Helpful to set in DNS beforehand
TCP/IP gateway	9.12.4.1	The router to and from the local subnet
DNS server 1	9.12.6.7	Assigned by the network administrator
DNS server 2/3 (optional)		Not used
OSA device name	eth0	Name of the interface to be assigned by IPWIZARD
OSA starting device number	3020	Start of OSA *triplet* for the z/VM TCP/IP stack
TCP/IP address	9.12.4.89	The TCP/IP address of the z/VM system
Subnet mask	255.255.252.0	Assigned by network administrator
OSA device type	QDIO	Often “QDIO” for OSA/Express cards
Network type	Ethernet	Usually “Ethernet”
Port name (optional)		Not required by z/VM
Router type	None	Usually “None”

Name	Value	Comment
Primary OSA device number for VSWITCH	3024	Specify the first device number (must be even number) and the next two device numbers will also be used
Secondary OSA device number for VSWITCH	3028	Should be on a different CHPID/OSA card
DASD addresses for z/VM	1A20 - 1A24	3390-3s to host z/VM
DASD addresses for Linux	1A25 - 1A29	3390-3s to host Linux guest
DASD addresses for paging	8228	3390-3s to host Linux swap space

Use the worksheet in Appendix B, “Planning worksheet” on page 409 to plan your z/VM installation resources.

Now that you have all of your resources planned for a z/VM and Linux installation, let’s get to do some real work.

Note: You should be familiar with some basic z/VM concepts. If you have not done so, please refer to 2.8, “Getting Started: The Basic Commands for z/VM” on page 70 and the previous sections of this chapter.

4.3.2 Installing z/VM from DVD

The section that follows assumes a first level installation of z/VM from DVD onto DASD. If you have not already done so, complete the worksheet in Appendix B, “Planning worksheet” on page 409. You will need access to the Hardware Management Console (HMC) with a user ID that has authority to go into single object operations mode.

z/VM 5.3 is shipped on tape and DVD. z/VM should install faster from tape due to faster I/O speeds, however, installing from tape may require more trips between the HMC and the tape drive.

If you are familiar with the HMC, you can use the two page *z/VM Summary for Automated Installation and Service (DVD Installation)* to replace or augment the description that follows.

If you are not familiar with the HMC and z/ VM, you may want to use the complete installation manual *z/VM Guide for Automated Installation and Service Version 5 Release 3.0*, GC24-6099. It is out of the scope of this book to describe how to set up an LPAR in the HMC for the system because there is no difference at all between creating an LPAR for z/VM and creating one for z/OS.

If you are installing z/VM at the *second level* (z/VM under z/VM) or onto SCSI disk, you will want to use the z/VM manual as the sections that follow do not address these options.

Installing z/VM from DVD

This section explains how to install z/VM 5.3 from an HMC with a DVD-ROM onto 3390-3 equivalent DASD. Some words are included for installing onto the larger 3390-9 DASD. For alternative configurations such as installing from tape or onto SCSI disks, refer to the z/VM documentation.

1. On the Hardware Management Console, select the LPAR on which you want to install z/VM.
2. If necessary, click the *racetrack* buttons (two buttons that are circular arrows on the bottom right corner) to traverse to the CPC Recovery (sometimes named just *Recovery*) menu.
3. On the CPC Recovery menu, double-click the **Integrated 3270 Console** as shown at the bottom of Figure 4-7. A window entitled *Integrated 3270 Console for <yourCPC>* will open (on older HMC levels, the window may be entitled *Personal Communications*).

> **Hint:** It is convenient to use the Alt-Tab key sequence to move between the HMC window and 3270 console.

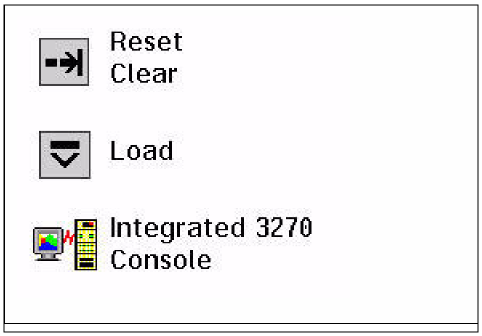

Figure 4-7 Integrated 3270 Console icon

4. Place the **z/VM Product Package Version 5 Release 3.0** DVD in the HMC DVD drive.
5. Get into *Single Object Operations* mode. To get into this mode, perform the following steps:

> **Note:** On the z10 you do not need to go to SE mode to load from DVD.

   a. Double-click the **Groups** icon in the *Views Area*
   b. Double-click **Defined CPCs** in the *Groups Work Area*.
   c. Select your CPC.
   d. If necessary, go around the *racetrack* (the buttons with circular arrows on the bottom right corner) to the *CPC Recovery* menu.
   e. Double-click the **Single Object Operations** icon. Click **yes** to confirm. Now a new window, *Primary Support Element Workplace™,* should appear (on older HMC levels it will be a “window within a window”). A window about a certificate may appear. If so, click **OK**.
   f. Double-click **Groups** near the top of this window.
   g. Double-click **Images** in the Groups Work Area.

> **Important:** If you are unable to get into Single Object Operations mode, it may be because you do not have sufficient permission. Check with the system administrator.

6. Select the LPAR that z/VM will be installed into.
7. Go around the racetrack in this window to the CPC Recovery menu. Double-click the **Load from CD-ROM or Server** icon when you see it (see Figure 4-8 on page 153).

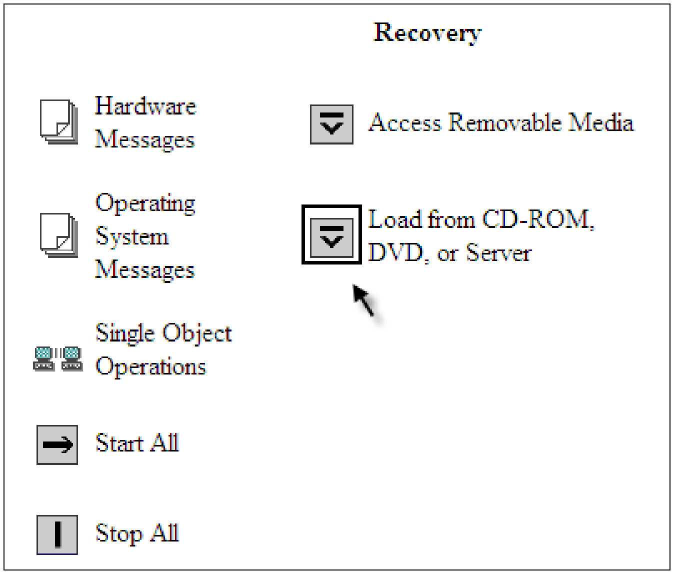

Figure 4-8 CPC Recovery menu with "Load from CD-ROM or Server" icon present.

8. On the *Load CD-ROM or Serve*r window as shown in Figure 4-9, the radio button **Hardware Management Console CD-ROM/DVD** should be selected.
9. In the same *Load CD-ROM or Serve*r window, fill in *File Location* with **/cpdvd**. This is the directory on the DVD with the z/VM 5.3 installation code.
10. Click **OK**.

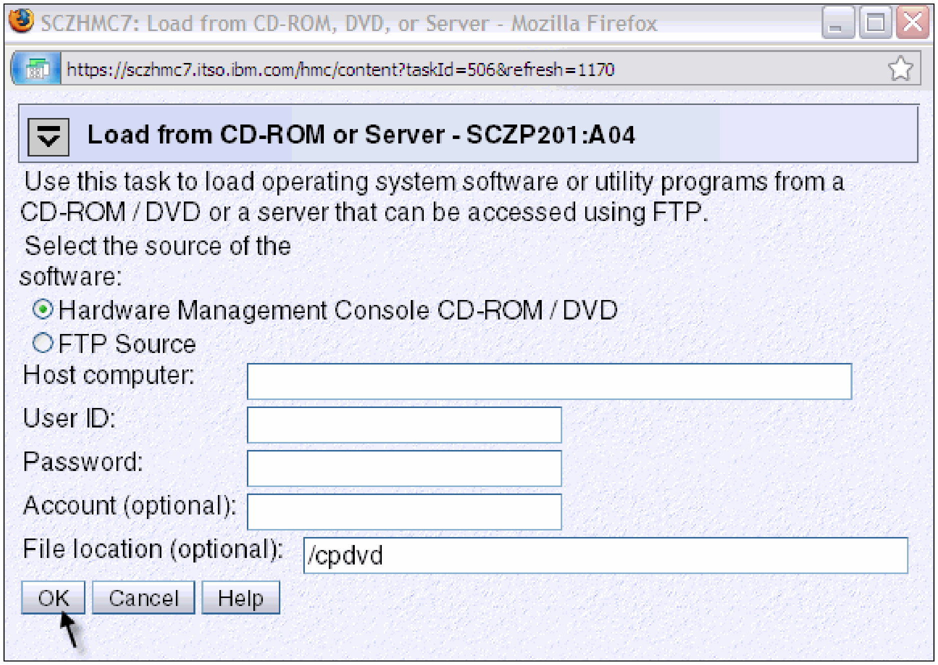

Figure 4-9 Load from CD-ROM or Server panel

11.Load the RAMDISK:

a. From the *Load from CD-ROM or Server* panel, the software **530vm.ins** should be selected as shown in Figure 4-10. Click **OK**.

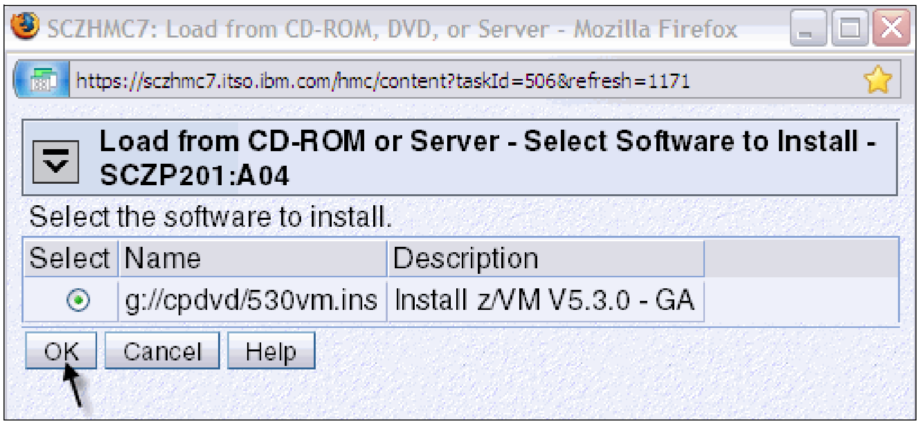

Figure 4-10 Selecting z/VM 5.3 RAMdisk system

b. From the Confirm the action window, click **Yes**. You should see the *Load from CD-ROM, DVD or Server Progress* window. The green light on the DVD drive should light up.

c. When you see the message `Completed successfully`. Click **OK** to close. This should normally take about four to six minutes.

> **Attention:** Normally, the z/VM RAMdisk (IBMVMRAM) loads in about four minutes. However, slow load times have been observed (15-18 minutes). When the green light on the DVD drive is solid, the load time will be acceptable. When it is intermittently dark more than it is green, long load times can result.

You should now have an in-memory z/VM 5.3 system running by now. The tasks performed in this section are similar to loading a Linux kernel onto memory for installation. You can compare this section with section 4.4.2, "Booting the Linux kernel for installation" on page 203 and later verify that both those systems load an image to main memory before the installation commences. A difference is that Linux starts the installation program automatically, whereas in z/VM you need to invoke a command as stated in the next section.

Copying a z/VM system image to DASD

This section describes the steps to copy z/VM to DASD.

1. You can now get out of *Single Object Operations* mode. To do so, log off the primary SE window by closing that window.
2. Use the **Alt-Tab** sequence, move to the Integrated 3270 Console window. The RAMdisk IPLs and the system comes up with the `MAINT user ID` logged on. You should see z/VM boot as shown in Figure 4-11 on page 156:

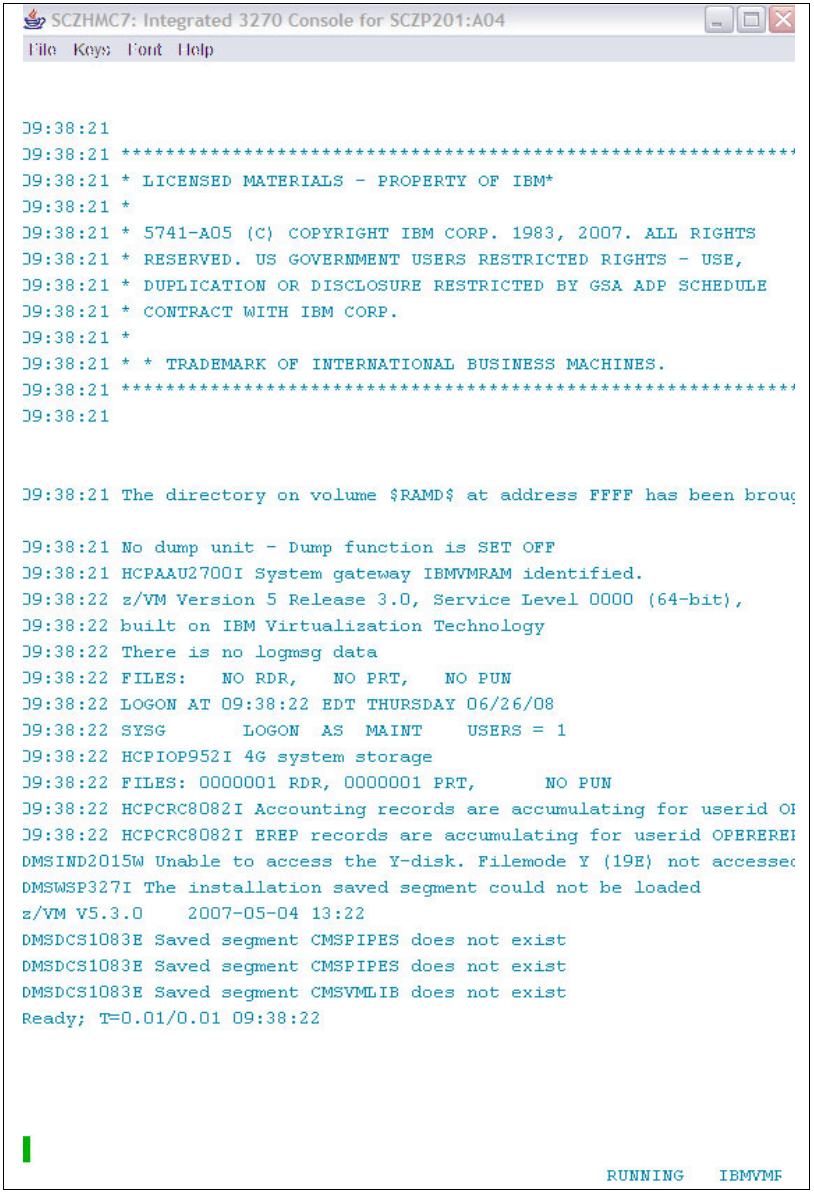

Figure 4-11 z/VM first boot on Integrated console

3. Invoke the **INSTPLAN** command. This will allow you to choose associated z/VM products to install, the language to use and the type of DASD on which to install:

```
==> instplan
```

You should see the display shown in Figure 4-12 It is recommended that you leave the M's in the top section alone.

Figure 4-12 Installation Planning menu

4. Type the letter **X** next to `AMENG` (or select your language) and `3390 Mod 3` (or the type of DASD you will use) as shown above.
5. Press **F5**. You should the message `HCPINP8392I INSTPLAN EXEC ENDED SUCCESSFULLY`.
6. Attach the DASD devices onto which z/VM will be installed defined in your planning worksheet in Appendix B., "Planning worksheet" on page 409. In this example, the devices are `1A20-1A24`.

```
==> att <1a20-1a24> *
1a20-1a24 ATTACHED TO MAINT
```

> **Important:** The angle brackets, <> , in the above example should not be typed. They are used throughout the book to mean that you should replace the example value with the correct value for your site. For example, if you are installing z/VM onto DASD 1200-1204, you would type the following:
>
> `==> att 1a00-1a04 *`

Running INSTDVD

The INSTDVD EXEC copies the z/VM system from DVD to disk.

1. Execute the **INSTDVD EXEC**:

   ```
   ==> instdvd
   ```

2. If you are using 3390-3s, you will see a panel asking for the five volumes as shown in Figure 4-13.

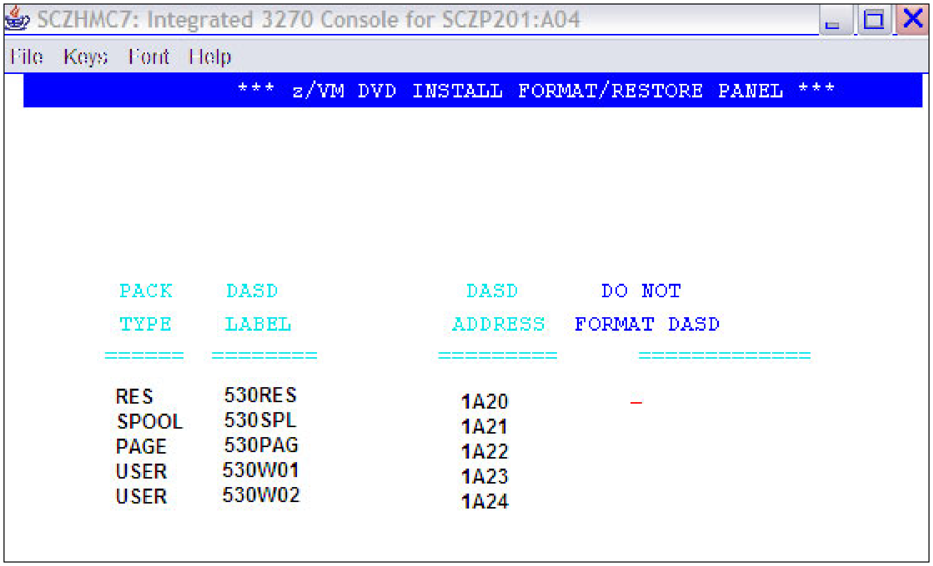

Figure 4-13 instdvd DASD screen

   a. Enter the addresses of the five volumes that z/VM will be installed on.

   b. Do *not* select the DO NOT FORMAT DASD check box on the right side of the panel.

   c. Notice that you can change the labels of all volumes but the 530RES one. We suggest that you change the other ones to something else other than

530. Section 4.3.3, “Relabeling the system volumes” on page 170 explains why you should change the labels. We changed ours to use the prefix *LX6*.

d. Press **F5** to start the installation.

3. When you see the question `DO YOU WANT TO CONTINUE?`, type **Y.** You should see the message `NOW FORMATTING DASD <1A20>`.

> **Important:** **`INSTDVD`** on a z10 LPAR took us about an hour.
>
> Read errors may be observed resulting in **`INSTDVD`** failing. If this is the case, you can try the command **`INSTDVD (RESTART`** and the install process should pick up where the read error occurred. This can be caused by dirt or fingerprints on the DVD.

4. You will be asked to place the system RSU in the drive. Insert the **z/VM Stacked Recommended Service Upgrade 5301** DVD into the HMC DVD-ROM drive and type **`GO`**. You should see a messages of the form `DVDLOAD: LOADING FILE CKD5000x IMAGE *.` This step takes two to four minutes.
5. Finally, you should see the message `HCPIDV8329I INSTDVD EXEC ENDED SUCCESSFULLY`.

IPL z/VM from DASD

IPL your initial z/VM system now on DASD.

1. From the HMC, **select your LPAR** by clicking it. You may have to first double-click **Groups**.
2. You should see the *CPC Recovery* (sometimes just *Recovery*) menu. Double-click the **Load** icon in the menu at the right side.
3. The Load window opens as shown in Figure 4-14. Follow these steps:
   a. Check the radio button **Clear**.
   b. Set the load address to the new system residence (530RES) volume which is **<1A20>** in this example.
   c. Set the load parameter to **SYSG**. This specifies to use the Integrated 3270 console.
   d. Click **OK** to IPL.

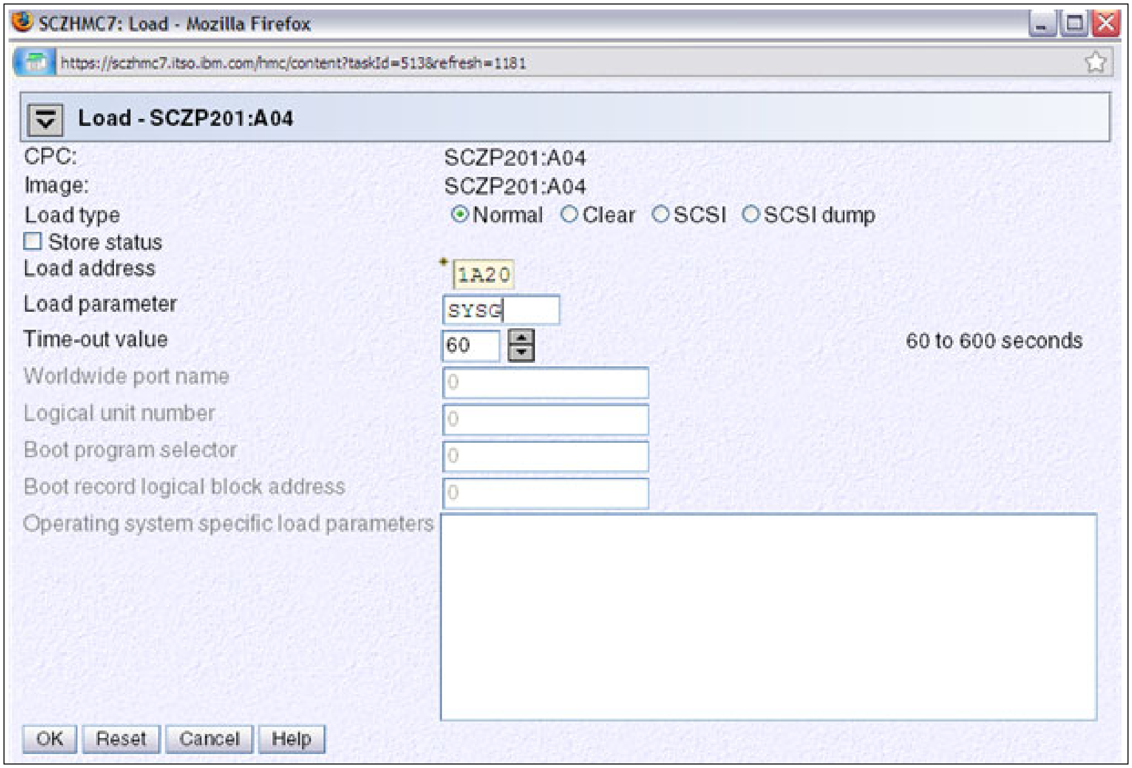

Figure 4-14 Load panel

4. When you see the *Load Task Confirmation* window, click **Yes**.
5. After 1-3 minutes you should see *Success* in the *Load Program* window. Click **OK**.
6. Use the **Alt-Tab** sequence to move back to the Integrated 3270 console window. You should see the *Standalone Program Loader* panel as shown in the following diagram.
   a. Press the **Tab** key to traverse to the *IPL Parameters* section (see Figure 4-15) and enter the value `cons=sysg`. This specifies the use of Integrated 3270 console.

```
STAND ALONE PROGRAM LOADER: z/VM VERSION 5 RELEASE 3.0

DEVICE NUMBER:   1A20       MINIDISK OFFSET: 00000000 EXTENT:1

 MODULE NAME:     CPLOAD     LOAD ORIGIN:       2000

 ----------------------IPL PARAMETERS-------------------------
cons=sysg

 -----------------------COMMENTS------------------------------



 -------------------------------------------------------------
```

Figure 4-15 IPL parameters.

b. Press the **F10** key to continue the IPL of your z/VM system. This should take around 2-3 minutes.

7. At the `Start (Warm|Force|COLD|CLEAN)` prompt, enter the following:

   `==>` **`cold drain noautolog`**

8. At the Change TOD clock prompt enter:

   `==>` **`no`**

9. The last message should be `HCPCRC8082I EREP records are accumulating for userID EREP`. Disconnect from the `OPERATOR` user ID via the **`DISCONNECT`** command (or its abbreviated form: **`DISC`**):

   `==>` **`disc`**

   Press **Enter** to get a new logon screen.

Completing the z/VM installation

Follow these steps to complete the z/VM installation

1. On the z/VM login screen, logon as **`MAINT`**. The password is **`MAINT`**. You may receive messages `HCPLMN108E` or `DMSACP113S` about disks not linked or attached. This is not a problem. Press **Enter** when you see the `VM Read` prompt in the lower right corner.

> **Important:** When logging onto a z/VM user ID that runs CMS, you should usually press **Enter** at the `VM READ` prompt. This will result in a prompt of the form:
>
> ```
> Ready; T=0.01/0.01 11:14:20
> ```

2. IPL CMS and press **Enter**. You should see the `Ready;` prompt.

   ```
   ==> ipl cms
   ==> Press Enter
   Ready;
   ```

3. Run the **`INSTVM DVD`** command:

   ```
   ==> instvm dvd
   ...
   HCPPLD8329I POSTLOAD EXEC ENDED SUCCESSFULLY
   ...
   HCPIVM8392I INSTVM ENDED SUCCESSFULLY
   ```

   This EXEC continues the installation process. This step should take about 4-8 minutes. The last message should be `HCPIVM8392I INSTVM ENDED SUCCESSFULLY`

4. Load the recommended service. For z/VM 5.3, the service name is **5301RSU1**. Run the following commands:

   ```
   ==> ipl cms
   ==> Press Enter
   Ready;
   ==> acc 500 c
   DMSACC724I 500 replaces C (2CC)
   ==> listfile * * c
   5301RSU1 SERVLINK C1
   ==> service all 5301rsu1
   ```

   This step should take about 3-6 minutes. The last message should be `VMFSRV2760I SERVICE processing completed successfully`.

5. Now IPL CMS and run the **`PUT2PROD`** command. This puts the service into production:

   ```
   ==> ipl cms
   ==> Press Enter
   Ready;
   ==> put2prod
   ```

   This step should take about 2-4 minutes. The last message should be `VMFP2P2760I PUT2PROD processing completed successfully`.

   A return code of 0 is ideal. You may get a return code of 4 and the message:

   ```
   VMFP2P2760I PUT2PROD process completed with warnings.
   ```

In general on z/VM, a return code of 4 is acceptable. That means that only warnings were issued. A return code of 8 or greater generally means that errors were encountered.

6. Enter the following command to shutdown and reIPL your system:

```
==> shutdown reipl
SYSTEM SHUTDOWN STARTED
```

7. You will lose your 3270 session. The system should come back in about 2-4 minutes. After it comes back, the last message should be “`Press enter or clear key to continue`”. **Press Enter** and you should see a z/VM logon screen.

Congratulations! You should now have a vanilla z/VM system installed.

Configuring the XEDIT profile

The **`Xedit`** command looks for the file `PROFILE XEDIT` configuration file when it is invoked. Many z/VM user IDs do not have such a personal or shared system file, so all `XEDIT` default values are in effect. The `MAINT 191` (A) disk has a `PROFILE XEDIT` so when you are editing files on `MAINT`, the values in this profile are usually in effect.

In order to have a good experience while using `XEDIT`, make its profile for the `MAINT` user be similar to the one found in `Example 2-25 on page 95`. To do so, logon to `MAINT` and edit the file with `XEDIT` itself:

```
==> xedit profile xedit a
```

Save your changes with the **`SAVE`** subcommand.

One default setting that can be dangerous, especially if you use `F12` to retrieve commands, is that PF12 is set to the **`FILE`** subcommand. Sometimes you may not want to save your changes with the stroke of one key. It is recommended that you set PF12 to the **`?`** subcommand which has the effect of a retrieve key:

```
==> x profile xedit a
```

Before:

```
SET PF12 FILE
```

After:

```
SET PF12 ?
```

Save your changes now and exit by using the **`FILE`** subcommand.

Configuring TCP/IP

It is recommended that you initially configure TCP/IP in your z/VM environment via the **IPWIZARD** command which is generally used just once. After **IPWIZARD** creates the initial configuration files, they are typically maintained manually.

From the z/VM logon panel **Logon to MAINT**. The default password for all z/VM user IDs is the same as the user ID. So enter a password of **maint** which will not be echoed on the screen.

```
USERID   ==> maint
PASSWORD ==>
```

After entering the user ID and password, press **Enter** when the status area in the lower right reads “VM READ”.

Use the IPWIZARD tool

The **IPWIZARD** command is on the MAINT 193 disk. You will need to access it file mode G via the **ACCESS** command so you will pick up **IPWIZARD** from that minidisk.

1. Access the MAINT 193 disk:

   ```
   ==> acc 193 g
   ```

2. Invoke **IPWIZARD**.

   ```
   ==> ipwizard
   ```

   ```
   *** z/VM TCP/IP Configuration Wizard ***

    The items that follow describe your z/VM host

    User ID of VM TCP/IP Stack Virtual Machine:   TCPIP___

    Host Name:     <vmlinux6>_____________
    Domain Name:    <itso.ibm.com>_________________________

    Gateway IP Address:  <9.12.4.1>________________________

    DNS Addresses:
    1) <9.12.6.7>_
    2) _______________
   ```

3. The *z/VM TCP/IP Configuration Wizard* opens as shown in the preceding example. The first field, User ID, should always be **TCPIP**. Obtain the remaining values from the , “Planning worksheets” on page 148 and press **F8**.

```
*** General Interface Configuration Panel ***

 Interface Name:   ETH0____________     Device Number:  <3020>

 IP Address:       <9.12.4.89>_
 Subnet Mask:      <255.255.252.0>__

 Interface Type (Select one):

    x    QDIO                  _     LCS                  _
HiperSockets
```

4. An *Interface Name* of **ETH0** is arbitrary but recommended. The *Device Number* will be the starting address of the OSA triplet that the z/VM stack will use. The *IP address* which must be routed to the OSA card will become the TCP/IP address of the z/VM system. The *Interface Type* will typically be **QDIO** with modern OSA devices. When completed, press **F8**.

```
*** QDIO Interface Configuration Panel ***

 Network Type (Select one):

    x    Ethernet             _     Token Ring


 Port Name (optional):  ________


 Router Type (Select one):

    _    Primary              _     Secondary           x    None
```

5. In general a value for the *Port Name* is no longer necessary[7] and a *Router Type* of **None** is recommended. Press **F5** to complete the wizard.

```
DTCIPW2508I DTCIPWIZ EXEC is attempting to create the necessary
DTCIPW2508I configuration files
```

6. Enter **1** to restart the TCP/IP stack:

[7] If the OSA card already has a port number defined, for example because it's being shared by a asystem that needs a port number on it, then you should set the same port number here.

```
The TCP/IP stack (TCPIP) must be restarted as part of this procedure
Would you like to restart and continue?
Enter 0 (No), 1 (Yes) 1
USER DSC   LOGOFF AS  TCPIP    USERS = 2     FORCED BY MAINT
...
Successfully PINGed Interface (9.12.4.89)
Successfully PINGed Gateway (9.12.4.1)
Ping Level 520: Pinging host 9.12.6.7
                Enter 'HX' followed by 'BEGIN' to interrupt.
```

Important: If the DNS server cannot be pinged, enter 1 to try it again.

```
PING: Ping #1 timed out
Not all of the PINGs were successful. Would you like
to try them again?
Enter 0 (No), 1 (Yes)
==> 1
...
```

Watch for the message **IPWIZARD EXEC ENDED SUCCESSFULLY**.

```
...
Successfully PINGed Interface (9.12.4.89)
Successfully PINGed Gateway (9.12.4.1)
Successfully PINGed DNS (9.12.6.7)
DTCIPW2519I Configuration complete; connectivity has been verified
DTCIPW2520I File PROFILE TCPIP created on TCPIP 198
DTCIPW2520I File TCPIP DATA created on TCPIP 592
DTCIPW2520I File SYSTEM DTCPARMS created on TCPIP 198
HCPINP8392I IPWIZARD EXEC ENDED SUCCESSFULLY
DMSVML2061I TCPIP 592 released
```

7. At this point your z/VM TCP/IP stack should be up. You should now be able to ping it from another system.

   If the **IPWIZARD** fails you must continue debugging it until it succeeds. Double check all values. Verify that the TCP/IP network and OSA information you were given are properly associated.

HMC Integrated 3270 Console or 3270 emulator? At this point z/VM should be accessible over the network. You can continue working at the HMC, or you can access your new system via a 3270 emulator.

If you want to switch to 3270 emulator, you can **LOGOFF**[a] of MAINT or **DISCONNECT**. If you logoff the session is ended. If you disconnect, your session remains where it is and is resumed when you log back on. In general, you should **LOGOFF** of system administration user IDs such as MAINT. However, you should always **DISCONNECT** from z/VM service machines such as TCPIP and user IDs running Linux. Logging off of them will terminate the service or crash Linux.

a. Abbreviated form: **LOG**

Configuring TCP/IP to start at IPL time

Bringing the TCP/IP stack automatically online during the system IPL is accomplished by including a few statements into AUTOLOG1's PROFILE EXEC file. As you may recall the discussion in "The AUTOLOG1 user" on page 134, this user is responsible for bringing other virtual machines online.

To accomplish that, follow these steps:

1. Logoff of MAINT.

```
==> log
```

2. You should see a new logon panel. **Logon to AUTOLOG1**. Again the password is the same as the user ID.

3. At the VM READ prompt enter the command **ACCESS (NOPROF** so that the PROFILE EXEC is not run.

```
LOGON AUTOLOG1
z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES:   NO RDR,   NO PRT,   NO PUN
LOGON AT 13:30:12 EST THURSDAY 11/29/07
DMSIND2015W Unable to access the Y-disk. Filemode Y (19E) not accessed
z/VM V5.3.0    2007-11-24 12:48
acc (noprof
```

4. Copy the PROFILE XEDIT from the MAINT 191 disk so XEDIT sessions will have a common interface among user IDs.

   a. Use the **VMLINK** command to both link to the disk read-only and to access it as the highest available file mode. The default read password is **read**:

   ```
   ==> vmlink maint 191
   ENTER READ PASSWORD:
   read
   ```

```
DMSVML2060I MAINT 191 linked as 0120 file mode Z
```

b. Copy the PROFILE XEDIT to your A disk:

```
==> copy profile xedit z = = a
```

5. Make a backup copy of the PROFILE EXEC and edit it:

```
==> copy profile exec a = execorig =
==> x profile exec
```

6. You should see the text in the top half of the following example. Modify it as follows.

   a. The z/VM Shared File System (SFS), is not required to run Linux so you can safely delete the three lines that **XAUTOLOG** the user IDs VMSERVS, VMSERVR and VMSERVU.

   b. You can also safely delete the Address Command line.

   c. Add a line to start the TCPIP user ID via the **XAUTOLOG** command and keep two statements that start the VSWITCH controllers.

   d. Add a line to **LOGOFF** of AUTOLOG1 when the EXEC is complete. There is no need to keep that virtual machine running as its sole purpose is to run the PROFILE EXEC.

Before:

```
/***************************/
/*  Autolog1 Profile Exec  */
/***************************/

Address Command
'CP XAUTOLOG VMSERVS'
'CP XAUTOLOG VMSERVU'
'CP XAUTOLOG VMSERVR'
```

After:

```
/***************************/
/*  Autolog1 Profile Exec  */
/***************************/
'cp xautolog tcpip'                    /* start up TCPIP */
'cp logoff'                            /* logoff when done */
```

7. Save your changes with the **FILE** subcommand and **LOGOFF** of AUTOLOG1:

```
====> file
==> log
```

When your z/VM system IPLs, the TCP/IP stack should now come up automatically (as long as you do *not* specify the notautolog parameter at IPL time).

Setting up an FTP server

In this section you learn how to set up an FTP server for your z/VM because you have to send a few files from your workstation to your Linux guest virtual machine later in section “Getting the files needed for linux boot” on page 193, and these files need to keep their 80-character record length property.

By default, z/VM has an FTP server included within its installation. To bring it online, log on as `TCPMAINT` and edit the `PROFILE TCPIP` file on its *198* disk, which is accessed as the *D* disk, and add an `AUTOLOG` statement near the top of the file with `FTPSERVE` as the only entry. In the `PORT` statement, remove the semicolons to uncomment the lines with `FTPSERVE` on them (ports 20 and 21). These changes will cause the FTP server to start when TCPIP is started. The important lines before the file is edited and after are shown here:

```
==> x profile tcpip d
```

Before:

```
; ----------------------------------------------------------------------
OBEY
OPERATOR TCPMAINT MAINT MPROUTE ROUTED DHCPD REXECD SNMPD SNMPQE
ENDOBEY
; ----------------------------------------------------------------------
PORT
; 20 TCP FTPSERVE  NOAUTOLOG ; FTP Server
; 21 TCP FTPSERVE            ; FTP Server
  23 TCP INTCLIEN            ; TELNET Server
; 25 TCP SMTP                ; SMTP Server
...
```

After:

```
; ----------------------------------------------------------------------
OBEY
OPERATOR TCPMAINT MAINT MPROUTE ROUTED DHCPD REXECD SNMPD SNMPQE
ENDOBEY
; ----------------------------------------------------------------------
AUTOLOG
  FTPSERVE  0
ENDAUTOLOG

PORT
  20   TCP FTPSERVE  NOAUTOLOG ; FTP Server
  21   TCP FTPSERVE            ; FTP Server
  23   TCP INTCLIEN            ; TELNET Server
; 25   TCP SMTP                ; SMTP Server
...
====> file
```

Save your changes with the **FILE** subcommand. These changes will take effect after you re-IPL your system.

Re-IPLing the system

You may want to be able to shutdown and reIPL z/VM without having to access the HMC. Often, the HMC will be logged off and thus the Integrated 3270 console (SYSG) will not be available. Because of these factors it is useful to use the *System Console* (SYSC - which has a title of *Operating System Messages* on the HMC) in order to shut down z/VM and reIPL it without needing to use the console. This console is always accessible whether you are logged on to the HMC or not. z/VM messages during both the shutdown and reIPL processes will be written to the system console, but often you will be able to ignore them - you just want your system back in a few minutes over the network.

It is now time to shutdown and reIPL the system. It is the fastest way to put the previous changes into effect since there is no production systems running yet. Pass the parameter **IPLPARMS CONS=SYSC** to the **SHUTDOWN REPIL** command:

```
==> shutdown reipl iplparms cons=sysc
```

You will lose your session, but it should come back in a few minutes.

If the network settings are correct, you can now access your z/VM through the network as opposed to using the HMC. You can use a 3270-like emulator, such as IBM Personnal communications, to access your network-enabled z/VM by its IP address.

4.3.3 Relabeling the system volumes

This step is optional, however, it is recommended. There are times when you will want to change the volume labels of the five z/VM system volumes (or three if you installed onto 3390-9s). If there is a possibility that another vanilla z/VM system with the same labels is installed onto volumes accessible by your z/VM system, one of the systems will not IPL correctly.

When installing z/VM it is possible to modify all but one volume label, that of the 530RES volume. This alleviates the problem that is described next, but it does not alleviate the problem of duplicate volume names.

To understand this possibility, refer to Figure 4-16 on page 171. The z/VM system with the lower device addresses starting at *E340* should IPL fine (though you may see a warning at system startup time about duplicate volume labels). However, if the z/VM system starting at device address F000 is IPLed, the 530RES volume will be used, but the remaining volumes in the system are searched for by volume label, not by device address. Because z/VM system 1's addresses are lower than z/VM system 2's, system 2 will be using system 1's volumes. This is not good for either system!

Linux systems that use label conventions to access their root partitions also suffer from a similar problem. If you install two Linux systems onto the same disk, both using labels to identify their partitions, the second system will actually use the first partition labelled as the root filesystem. Using labels in Linux is not recommended by the authors of this book.

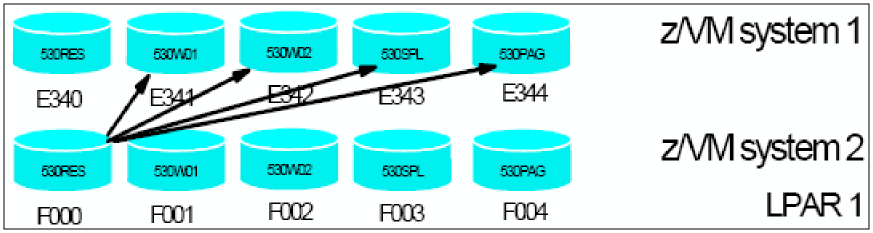

Figure 4-16 The problem with two z/VM systems with identical volume labels

If there is a possibility of another z/VM system being installed on DASD that this system will have access to, it is recommended that you perform the following steps. You will need access to the HMC to perform them:

- "Modifying labels in the SYSTEM CONFIG file" on page 171
- "Modifying labels in the USER DIRECT file" on page 173
- "Changing the labels on the five volumes" on page 174
- "Shutting the system down and reipling it" on page 175

Important: This process must be done as documented. Making a mistake in one of the steps can easily result in an unusable system. Check your steps carefully and your system will come back with no problems. Try to do all steps in succession in a short amount of time. Close your door, don't answer your phone or e-mail, turn off instant messaging :))

Modifying labels in the SYSTEM CONFIG file

This file controls what DASDs are used by the system, what virtual resources are defined and so on. Here, you will modify some of the DASD settings to allow the system to correctly start up with the new labels.

An HMC 3270 session is needed because z/VM will have to be restarted with a `FORCE` option.

1. Start an **Integrated 3270 Console** session on the HMC from the *CPC Recovery* (or just *Recovery*) menu.
2. Note the first five CP-owned volumes via the **`QUERY CPOWNED`** command:

```
==> q cpowned
```

```
Slot  Vol-ID  Rdev  Type   Status
   1  530RES  A700  Own    Online and attached
   2  LX6SPL  A701  Own    Online and attached
   3  LX6PAG  A702  Own    Online and attached
   4  LX6W01  A703  Own    Online and attached
   5  LX6W02  A704  Own    Online and attached
   6  MPA705  A705  Own    Online and attached
   7  MPA706  A706  Own    Online and attached
   8  MPA707  A707  Own    Online and attached
   9  MPA708  A708  Own    Online and attached
  10  MPA709  A709  Own    Online and attached
  11  ------  ----  -----  Reserved
  12  ------  ----  -----  Reserved
 ...
```

Note that, since we had changed the labels for the other four DASDs in "Running INSTDVD" on page 158, only the 530RES DASD needs to have its label changed.

For 3390-9s: If z/VM is installed onto 3390-9s, there should only be three system volumes:

```
==> q cpowned
Slot  Vol-ID  Rdev  Type   Status
   1  530RES  9300  Own    Online and attached
   2  530SPL  9301  Own    Online and attached
   3  530PAG  9302  Own    Online and attached
 ...
```

3. An XEDIT sub-command is used to help make this process more reliable. It can be used on both the SYSTEM CONFIG and USER DIRECT files. To modify the labels in the SYSTEM CONFIG file, release the A CP-disk and access it read-write. Back up the SYSTEM CONFIG file:

```
==> cprelease a
CPRELEASE request for disk A scheduled.
HCPZAC6730I CPRELEASE request for disk A completed.
==> link * cf1 cf1 mr
==> acc cf1 f
==> copy system config f = confwrks = (oldd rep
```

4. Edit the SYSTEM CONFIG file and use the **ch** sub-command with each label you need to relabel (530RES only, in this example):

```
==> xedit system config f
==> ch /530RES/<LX6RES>/* *
```

Note: Don't forget to change <LX6RES> to whichever prefix you are using.

After each change command, you should see a message on the screen indicating that lines have been changed.

5. Clear the screen and you will be left in XEDIT editing the file. Search for the string cp_owned and you should see the new labels. Be sure they are correct before saving the file with the **FILE** subcommand:

```
====> /cp_owned
/*                     CP_Owned Volume Statements                     */
/**********************************************************************/

 CP_Owned   Slot   1  LX6RES
 CP_Owned   Slot   2  LX6SPL
 CP_Owned   Slot   3  LX6PAG
 CP_Owned   Slot   4  LX6W01
 CP_Owned   Slot   5  LX6W02
    ...
====> file
```

6. Verify there are no syntax errors:

```
==> acc 193 g
==> cpsyntax system config f
CONFIGURATION FILE PROCESSING COMPLETE -- NO ERRORS ENCOUNTERED.
```

7. Release and detach the F disk, **CPACCESS** (or its abbreviation: **CPACC**) the A disk and verify:

```
==> rel f (det
DASD 0CF1 DETACHED
==> cpacc * cf1 a
CPACCESS request for mode A scheduled.
Ready; T=0.01/0.01 09:19:57
HCPZAC6732I CPACCESS request for MAINT's 0CF1 in mode A completed.
==> q cpdisk
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  530RES A700 CKD          39         83
MNTCF2 MAINT    0CF2  B   R/O  530RES A700 CKD          84        128
MNTCF3 MAINT    0CF3  C   R/O  530RES A700 CKD         129        188
```

You have now changed the labels of the system volumes in the SYSTEM CONFIG file. It is critical that you proceed as your system is now in a state where it will not IPL cleanly.

Modifying labels in the USER DIRECT file

In this section you will modify the system volume labels in the USER DIRECT file. The USER DIRECT file is used to store information about the space used by each z/VM virtual machine, that is, each user. The changes made to DASD labels need to be reflected here also, or your system will end up in an inconsistant state.

1. Modify the `USER DIRECT` file again using XEDIT's **ch** sub-command. You should see many more occurrences of the labels being changed:

```
==> copy user direct c = direwrks = (oldd rep
==> x user direct c
===> ch /530RES/<LX6RES>/* *
DMSXCG517I 84 occurrence(s) changed on 84 line(s)
```

   You may choose to traverse the file to verify it before saving the changes with the **FILE** subcommand:

```
====> file
```

You have now changed the labels of the system volumes in the USER DIRECT and `SYSTEM CONFIG` files. Again, it is critical that you proceed with the remaining steps.

Changing the labels on the five volumes

To relabel a pack, you need write access to cylinder 0 of the DASD volume.There are already mdisk statements that give you that access.

1. You can query the disks to make sure you have R/W access to them:

```
query virtual 122-125
   DASD 0122 3390 LX6SPL R/W 3339 CYL ON DASD  1A21 SUBCHANNEL = 000A
   DASD 0123 3390 530RES R/W 3339 CYL ON DASD  1A20 SUBCHANNEL = 000B
   DASD 0124 3390 LX6W01 R/W 3339 CYL ON DASD  1A23 SUBCHANNEL = 000C
   DASD 0125 3390 LX6W02 R/W 3339 CYL ON DASD  1A24 SUBCHANNEL = 000D
```

Note: The mapping of virtual addresses and volumes defaults as shown below:

- MAINT 123 for 530RES
- MAINT 122 for 530SPL
- MAINT 124 for 530W01
- MAINT 125 for 530w02
- $PAGE$ A03 for 530PAG

You may have to link to the paging minidisk before querying it, if you also have to relabel it (which is not the case of our example installation since it's already properly labelled):

```
link $page$ a03 a03 mr
Ready; T=0.01/0.01 17:24:51
query virtual a03
DASD 0A03 3390 LX6PAG R/W 3339 CYL ON DASD  1A22 SUBCHANNEL = 004A
```

2. Relabel each of the packs with the appropriate new label.

   ```
   cpfmtxa 123 <LX6RES> label
   ```

   Note: Make sure you enter the last parameter - **label** on the command.

   The z/VM system does not reread the pack labels until the next IPL, so if you were to display these packs, the old labels will still displayed.

3. Load the updated USER DIRECT file to the system directory space.

   ```
   directxa user direct c
   ```

4. Once the new directory is loaded, you can shutdown and reIPL the system.

Shutting the system down and reipling it

You will need an HMC console session for this step, if you are not already running from there.

To test the changes you must shut your system down and then restart it. You cannot do a `SHUTDOWN REIPL` in this situation because you will have to do a FORCE start

```
==> shutdown
SYSTEM SHUTDOWN STARTED
HCPSHU960I System shutdown may be delayed for up to 210 seconds
```

Perform the following steps to bring the system back up:

1. From the HMC click the LOAD icon in the CPC Recovery (or just *Recovery*) menu.
2. Select the **Clear** radio button. All the other parameters should be correct from the previous IPL. Click **OK**
3. Click **Yes** on the *Load Task Confirmation* panel.
4. Go back to the Integrated 3270 console. After a few minutes the *Standalone Program Loader* panel should appear. Use the **TAB** key to traverse to the section *IPL Parameters* and enter the value **`cons=sysg`**
5. Press the **F10** key to continue the IPL of your z/VM system. This takes around 3 minutes.
6. At the Start prompt you have to specify a FORCE start, again because the spool volume label has changed. Enter the following:

   ```
   ==> force drain
   ```

7. Do not change the time of day clock:

   ```
   ==> no
   ```

8. When the IPL completes, **DISCONNECT** from the OPERATOR user ID and **logon to** MAINT:

```
==> disc
```

You should now be able to get a 3270 emulator session as the TCPIP service machine should be up. Get a 3270 session as MAINT and verify the volume labels have changed with the **QUERY CPOWNED** command:

```
==> q cpowned
Slot  Vol-ID  Rdev  Type   Status
   1  LX6RES 1A20  Own    Online and attached
   2  LX6SPL 1A21  Own    Online and attached
   3  LX6PAG 1A22  Own    Online and attached
   4  LX6W01 1A23  Own    Online and attached
   5  LX6W02 1A24  Own    Online and attached
  ...
```

In the event that you IPLed a system with duplicate system volumes, it is possible that you may have destroyed your saved segments. You will know this is the case when you cannot **IPL CMS**. Rather, you will have to **IPL 190**.

Important: Only do this if your saved segments have been destroyed! To rebuild saved segments, try the following commands, as MAINT:

```
==> vmfsetup zvm cms
==> sampnss cms
==> i 190 cl parm savesys cms
==> vmfbld ppf segbld esasegs segblist ( all
```

4.4 Installing Linux

4.4.1 Configuring your z/VM system for installing Linux

Having a simple but efficient environment to run Linux virtual machines on z/VM requires that you first configure and customize your environment so that resources such as DASDs, networking and easy-to-manage-and-install Linux guests are all available at the time you hit the red button to execute the Linux kernel. The tasks involved here are:

- Customizing the SYSTEM CONFIG file
  - Change some of its parameters
  - Create a virtual switch
- Setting up the DASDs for the Linux guest

  – Formatting the DASDs
  – Configure DASDs for system's and user's use
- Creating the Linux Maintenence User
  – Create minidisks
  – Set up a common environment for booting Linux in guest virtual machines

Recall section 4.2.2, "System files" on page 132 where we mentioned some important ocnfiguration files of z/VM. You are about to customize them now.

Customizing the SYSTEM CONFIG file

The first configuration file read when z/VM IPLs is the `SYSTEM CONFIG` file. The following changes are recommended:

- Increase retrieve key capacity
- Allow virtual disks (VDISKs) to be created
- Turn off the Disconnect Timeout. This will prevent idle disconnected users from being forced off the system
- Define a virtual switch (VSWITCH) that will be used for Linux networking
  – Defining a virtual switch for all of the Linux guest systems is a good practice because you will probably not have enough physical OSA network cards for every guest virtual machine that you create if you are planning to grow to many ones later on. A virtual switch will use a single real OSA device and will share network connectivity among multiple virtual guests.

To make these changes, perform the following steps:

1. To edit the `SYSTEM CONFIG` file, the `MAINT CF1` minidisk must be released as a CP disk via the **CPRELEASE** (or its abbreviation: **CPREL**) command. The CP disks are queried via the **QUERY CPDISK** command. Note that, since the *CF1* disk is owned by CP, it's being accessed in read only mode by `MAINT` as CP disk A. So, in order to access that CF1 disk in multi-read mode, we have to request CP to release it first, and later access it in multi-read (*mr*) mode.

```
==> q cpdisk
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  MVA740 A740 CKD          39        158
MNTCF2 MAINT    0CF2  B   R/O  MVA740 A740 CKD         159        278
MNTCF3 MAINT    0CF3  C   R/O  MVA740 A740 CKD         279        398
==> cprel a
CPRELEASE request for disk A scheduled.
HCPZAC6730I CPRELEASE request for disk A completed.
==> q cpdisk
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF2 MAINT    0CF2  B   R/O  MVA740 A740 CKD         159        278
MNTCF3 MAINT    0CF3  C   R/O  MVA740 A740 CKD         279        398
```

2. Once it is released, you are able to access the MAINT CF1 disk in read-write mode. Use the **LINK** command with multi-read (**MR**) parameter and **ACCESS** command to get read-write access to the minidisk.

```
==> link * cf1 cf1 mr
==> acc cf1 f
```

3. Now the MAINT CF1 disk is accessed read-write as your F disk. First make a backup copy of the vanilla SYSTEM CONFIG file using the COPYfile command with the **OLDDATE** parameter so the timestamp of the file is not modified, then edit the original copy:

```
==> copy system config f system conforig f (oldd
==> x system config f
```

4. Next look for the Features statement. You can search for it again or you can use **F8** to page down. The following changes and additions are recommended:

 – Increase the number of commands that can be retrieved from 20 to 99.
 – Set the Disconnect_Timeout to **off** so disconnected users do not get forced off.
 – Allow unlimited VDISKs to be created by users by changing Userlim to **infinite** and by adding the **Syslim infinite** clause:

```
Features ,
  Disable ,                          /*Disable the following features*/
    Set_Privclass ,                  /*Disallow SET PRIVCLASS command*/
    Auto_Warm_IPL ,                  /*Prompt at IPL always */
    Clear_TDisk   ,                  /*Don't clear TDisks at IPL time*/
  Retrieve ,                         /*Retrieve options */
    Default  99 ,                    /*Default.... default is 20 */
    Maximum  255 ,                   /*Maximum.... default is 255 */
  MaxUsers noLimit ,                 /*No limit on number of users */
  Passwords_on_Cmds ,                /*What commands allow passwords?*/
    Autolog  yes ,                   /*... AUTOLOG does */
    Link     yes ,                   /*... LINK does */
    Logon    yes ,                   /*... and LOGON does, too */
  Disconnect_Timeout off ,           /*Don't force disconnected users*/
  Vdisk ,                            /*Allow VDISKS for Linux swaps   */
    Syslim  infinite  ,
    Userlim  infinite
```

> **Note:** If you want to limit the number of VDISKs user can create, you should pick up a limit for it instead of using *infinite*. More about VDISKS in “Virtual DASD (VDISK)” on page 265.

5. Define a VSWITCH:

Use the **BOTTOM** (or its abbreviation BOT) subcommand to go to the bottom of the file. Add some lines (you can use the XEDIT add subcommand **a3**). Define a VSWITCH and set the MAC address prefix. If you have multiple z/VM systems, each should have a unique prefix. Modify the two starting addresses of the OSA triplets (3024 and 3028 in this example) to those you specified in , “Planning worksheets” on page 148.

```
====> bot
====> a3
/* define vswitch named vsw1 and set MAC address prefixes to
02-00-01 */
define vswitch vsw1 rdev <3024> <3028>
vmlan macprefix 020001
```

6. Save your changes with the XEDIT **FILE** subcommand:

```
====> file
```

7. Test your changes with the **CPSYNTAX** command which is on the MAINT 193 disk:

```
==> acc 193 g
==> cpsyntax system config f
CONFIGURATION FILE PROCESSING COMPLETE -- NO ERRORS ENCOUNTERED.
```

Pay attention to the output. If you get any syntax errors, fix them before proceeding.

8. Release and detach the MAINT CF1 disk with the **RELEASE** command (abbreviated as **REL**) and the **DETACH** parameter. Then put it back online with the **CPACCESS** command:

```
==> rel f (det
DASD 0CF1 DETACHED
==> cpacc * cf1 a
CPACCESS request for mode A scheduled.
HCPZAC6732I CPACCESS request for MAINT's 0CF1 in mode A completed.
==> q cpdisk
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  MVA740 A740 CKD          39        158
MNTCF2 MAINT    0CF2  B   R/O  MVA740 A740 CKD         159        278
MNTCF3 MAINT    0CF3  C   R/O  MVA740 A740 CKD         279        398
```

Note that all three CP disks are now accessed.

Starting virtual switches automatically at system startup

As discussed in “Configuring TCP/IP to start at IPL time” on page 167, the AUTOLOG1 PROFILE EXEC is the best place to auto start virtual resources during the system IPL.Therefore, the virtual switches should be started by that profile. In addition to that, it is recommended that the following tasks be accomplished by using `AUTOLOG1`s `PROFILE EXEC`:

- Configure Linux to shut down gracefully via the SET SIGNAL command
- Overcommit memory via the SET SRM STORBUF command
- Grant access to the VSWITCH for each Linux user
- Limit minidisk cache in central storage and turn it off in expanded storage

To accomplish those, do the following:

1. **Logon** to `AUTOLOG1`. At the `VM READ` prompt you have usually been pressing Enter which causes the `PROFILE EXEC` to be run. If you do not want this `EXEC` to run, enter the command **`ACCESS (NOPROF`**:

```
LOGON AUTOLOG1
z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
built on IBM Virtualization Technology
There is no logmsg data
FILES:   NO RDR,   NO PRT,   NO PUN
LOGON AT 13:39:10 EST SUNDAY 11/24/07
DMSIND2015W Unable to access the Y-disk. Filemode Y (19E) not
accessed
z/VM V5.3.0    2007-11-10 09:57
==> acc (noprof
```

2. Make a copy of the working `PROFILE EXEC`

```
==> copy profile exec a = execwrks =
```

3. Edit the file and add the emboldened text.

```
==> x profile exec
/***************************/
/*  Autolog1 Profile Exec  */
/***************************/
'cp xautolog tcpip'              /* start up TCPIP */
'CP XAUTOLOG DTCVSW1'            /* start VSWITCH controller 1 */
'CP XAUTOLOG DTCVSW2'            /* start VSWITCH controller 2 */
'cp set pf12 ret'                /* set the retrieve key */
'cp set mdc stor 0m 128m'        /* Limit minidisk cache in CSTOR */
'cp set mdc xstore 0m 0m'        /* Disable minidisk cache in XSTOR */
'cp set srm storbuf 300% 250% 200%' /* Overcommit memory */
'cp set signal shutdown 180'     /* Allow guests 3 min to shut down */

'cp logoff'                       /* logoff when done */
```

Save your changes with the FILE subcommand.

Important: The **set mdc** and **set srm** lines are z/VM tuning values. It is believed that these are good starts for Linux systems, but may not be optimal. For more reading on these values see the following Web sites:

http://www.vm.ibm.com/perf/tips/linuxper.html
http://www.vm.ibm.com/perf/tips/prgmdcar.html

Note: You can also put the changes above into the SYSTEM CONFIG file.

You may choose to modify or omit some of these settings. Your system should now be configured to start up the virtual switches and send a signal to shut down Linux virtual machines gracefully.

Verifying the changes

It is now time to make these changes take effect. You can do that by re-IPLing the system again or by issuing the commands you just placed into AUTOLOG1's profile. If you are going to re-IPL, pass the parameter **IPLPARMS CONS=SYSC** to the **SHUTDOWN REPIL** command:

```
==> shutdown reipl iplparms cons=sysc
```

You will lose your session, but it should come back in a few minutes. When you are done with this step, perform the following commands:

1. Start a 3270 session and **Logon** as MAINT.
2. Query the new VSWITCH:

```
==> q vswitch
VSWITCH SYSTEM VSW1     Type: VSWITCH Connected: 0    Maxconn: INFINITE
  PERSISTENT  RESTRICTED    NONROUTER                 Accounting: OFF
  VLAN Unaware
  State: Ready
  IPTimeout: 5         QueueStorage: 8
  Portname: UNASSIGNED RDEV: 3024 Controller: DTCVSW1  VDEV:  3024
  Portname: UNASSIGNED RDEV: 3028 Controller: DTCVSW2  VDEV:  3028 BACKUP
```

   You should see that the VSWITCH exists and that there are two built-in VSWITCH controllers, DTCVSW1 and DTCVSW2. Before z/VM 5.2, these user IDs had to be created manually.

3. Use the **Query VDISK** and **Query RETRIEVE** commands to see the changes made to the Features statement in the SYSTEM CONFIG file:

```
==> q retrieve
99 buffers available.  Maximum of 255 buffers may be selected.
==> q vdisk userlim
VDISK USER   LIMIT IS INFINITE
==> q vdisk syslim
```

```
VDISK SYSTEM LIMIT IS INFINITE,               0 BLK IN USE
```

This shows that the changes to the SYSTEM CONFIG file have taken effect.

Setting up DASDs

In this step you will bring online your DASDs reserved for the Linux guest. There are two steps: formatting them and registering them into the system.

You should only format the extra DASDs that you will use for system paging. The other DASDs intended to host the guest operating system (Linux) will be formatted by the guest operating system itself later.

Formatting the DASDs

Let's format the DASDs we have planned to use as paging. This is done through the **CPFMTXA** command, which is an interactive program to format disks. Follow the steps below to format your disks.

> **z/OS anology:** The analogous z/OS command for this would be to run a batch job executing the program ICKDSF using the appropriate parameters.
>
> In Linux, the analogous command is **mkfs**.

1. Use the **ATTACH** command (abbreviated as **ATT**) to attach the DASDs to our user space so that you can see them:

   Example 4-2 Attaching DASDs to the system

   ```
   att 8228 *
   DASD 8228 ATTACHED TO MAINT 8228 WITH DEVCTL
   ```

2. Now use the **CPFMTXA** command on each of the disks to format them. Remember to format themas PAGE volumes. Example 4-3 shows the interaction within the **CPFMTXA** command to format our PAGE DASD. User inputs are in **bold**.

Example 4-3 Formatting disks with cpfmtxa

```
cpfmtxa
ENTER FORMAT, ALLOCATE, LABEL, OR QUIT:
format
ENTER THE VDEV TO BE PROCESSED OR QUIT:
8228
ENTER THE CYLINDER RANGE TO BE FORMATTED ON DISK 8228 OR QUIT:
0-end
ENTER THE VOLUME LABEL FOR DISK 8228:
```

```
LX6PG2
CPFMTXA:
FORMAT WILL ERASE CYLINDERS 00000-03338 ON DISK 8228
DO YOU WANT TO CONTINUE? (YES | NO)
yes
HCPCCF6209I INVOKING ICKDSF.
ICK030E DEFINE INPUT  DEVICE: FN FT FM, "CONSOLE", OR "READER"
CONSOLE
ICK031E DEFINE OUTPUT DEVICE: FN FT FM, "CONSOLE", OR "PRINTER"
CONSOLE
ICKDSF - CMS/XA/ESA DEVICE SUPPORT FACILITIES 17.0                TIME:
14:09:51
       05/14/08     PAGE   1

ENTER INPUT COMMAND:
 CPVOL FMT MODE(ESA) UNIT(8228) VOLID(DK8228) NOVFY -
ENTER INPUT COMMAND:
 RANGE(0,3338)
ICK00700I DEVICE INFORMATION FOR 8228 IS CURRENTLY AS FOLLOWS:
         PHYSICAL DEVICE = 3390
         STORAGE CONTROLLER = 3990
         STORAGE CONTROL DESCRIPTOR = E9
         DEVICE DESCRIPTOR = 0A
         ADDITIONAL DEVICE INFORMATION = 4A001B35
         TRKS/CYL = 15, # PRIMARY CYLS = 3339
ICK04000I DEVICE IS IN SIMPLEX STATE
ICK00091I 8228 NED=002105.000.IBM.13.000000022513
ICK091I 8228 NED=002105.000.IBM.13.000000022513
ICK03020I CPVOL WILL PROCESS 8228 FOR VM/ESA MODE
ICK03090I VOLUME SERIAL = DK8228
ICK03022I FORMATTING THE DEVICE WITHOUT FILLER RECORDS
ICK03011I CYLINDER RANGE TO BE FORMATTED IS 0 - 3338
ICK003D REPLY U TO ALTER VOLUME 8228 CONTENTS, ELSE T
U
ICK03000I CPVOL REPORT FOR 8228 FOLLOWS:

        FORMATTING OF CYLINDER 0 STARTED AT: 14:09:51
        FORMATTING OF CYLINDER 100 ENDED AT: 14:09:54
        FORMATTING OF CYLINDER 200 ENDED AT: 14:09:56
        ....
        FORMATTING OF CYLINDER 3100 ENDED AT: 14:12:07
        FORMATTING OF CYLINDER 3200 ENDED AT: 14:12:10
        FORMATTING OF CYLINDER 3300 ENDED AT: 14:12:13
        FORMATTING OF CYLINDER 3338 ENDED AT: 14:12:14
```

```
          VOLUME SERIAL NUMBER IS NOW = LX6PG2

        CYLINDER ALLOCATION CURRENTLY IS AS FOLLOWS:
TYPE      START      END        TOTAL
         ----     -----      ---        -----
         PERM     0          3338       3339

ICK00001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
         14:12:14    05/14/08

ENTER INPUT COMMAND:
 END

ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
ENTER ALLOCATION DATA
TYPE CYLINDERS
.................
page 1-end
end
HCPCCF6209I INVOKING ICKDSF.
ICK030E DEFINE INPUT  DEVICE: FN FT FM, "CONSOLE", OR "READER"
CONSOLE
ICK031E DEFINE OUTPUT DEVICE: FN FT FM, "CONSOLE", OR "PRINTER"
CONSOLE
ICKDSF - CMS/XA/ESA DEVICE SUPPORT FACILITIES 17.0                TIME:
14:14:07
       05/14/08     PAGE   1

ENTER INPUT COMMAND:
  CPVOL ALLOC MODE(ESA) UNIT(8228) VFY(LX6PG2) -
ENTER INPUT COMMAND:
       TYPE((PERM,0,3338))
ICK00700I DEVICE INFORMATION FOR 8228 IS CURRENTLY AS FOLLOWS:
         PHYSICAL DEVICE = 3390
         STORAGE CONTROLLER = 3990
         STORAGE CONTROL DESCRIPTOR = E9
         DEVICE DESCRIPTOR = 0A
        ADDITIONAL DEVICE INFORMATION = 4A001B35
         TRKS/CYL = 15, # PRIMARY CYLS = 3339
ICK04000I DEVICE IS IN SIMPLEX STATE
ICK00091I 8228 NED=002105.000.IBM.13.000000022513
ICK091I 8228 NED=002105.000.IBM.13.000000022513

ICK03020I CPVOL WILL PROCESS 8228 FOR VM/ESA MODE
ICK03090I VOLUME SERIAL = LX6PG2
```

```
ICK03024I DEVICE IS CURRENTLY FORMATTED WITHOUT FILLER RECORDS
ICK003D REPLY U TO ALTER VOLUME 8228 CONTENTS, ELSE T
U
ICK03000I CPVOL REPORT FOR 8228 FOLLOWS:

        CYLINDER ALLOCATION CURRENTLY IS AS FOLLOWS:
         TYPE     START      END        TOTAL
         ----     -----      ---        -----
          PAGE         1     3338        3338

ICK00001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0
         14:14:42    05/14/08

ENTER INPUT COMMAND:
 END

ICK00002I ICKDSF PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
Ready; T=0.06/0.15 14:14:42
```

Repeat this process for every disk.

Registering the disks into the system

Once a DASD is formatted, we want to make sure that the system recognizes them as available storage. This information is stored in the `SYSTEM CONFIG` file. This would be analogous to (assuming the device is defined in the HCD IOCDS) issuing a **VARY xxxx,ONLINE** and eventually put that command in `COMMNDxx` or `IEACMD00` of `PARMLIB` concatenation if defined with `OFFLINE` attribute in z/OS (any mount usage attributes would also have to be considered). Linux systems don't have an equivalent step since disks are discovered in boot time by a hardware probing program and automatically appear as devices under `/dev` after the system finishes booting up.

Follow these steps to update the `SYSTEM CONFIG` file:

1. The following example uses the same steps to access the `MAINT CF1 disk` read-write that you used earlier:

```
==> q cpdisk
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  530RES 0200 CKD          39         83
MNTCF2 MAINT    0CF2  B   R/O  530RES 0200 CKD          84        128
MNTCF3 MAINT    0CF3  C   R/O  530RES 0200 CKD         129        188
==> cprel a
CPRELEASE request for disk A scheduled.
HCPZAC6730I CPRELEASE request for disk A completed.
==> link * cf1 cf1 mr
```

```
==> acc cf1 f
```

It is good to remember this sequence of steps.

2. Edit the SYSTEM CONFIG file and specify each of the new page volumes (PAGE) by name as CP_Owned. When you system IPLs it will pick up these as paging volumes.

```
==> x system config f
...
/*****************************************************************/
/*                    CP_Owned Volume Statements                 */
/*****************************************************************/

  CP_Owned   Slot   1  LX6RES
  CP_Owned   Slot   2  LX6SPL
  CP_Owned   Slot   3  LX6PAG
  CP_Owned   Slot   4  LX6W01
  CP_Owned   Slot   5  LX6W02
  CP_Owned   Slot   6  LX6PG2
    CP_Owned   Slot   7  RESERVED
    CP_Owned   Slot   8  RESERVED
    CP_Owned   Slot   9  RESERVED
    CP_Owned   Slot   10  RESERVED
    CP_Owned   Slot   11  RESERVED
    CP_Owned   Slot   12  RESERVED
    CP_Owned   Slot   13  RESERVED
...
```

3. Move down to the User_Volume_List section. User volumes (PERM) can be specified individually with the User_Volume_List statement, or with wild cards via the User_Volume_Include statement. Now add the user volumes which will be used by the Linux guest:

```
/*****************************************************************/
/*                          User_Volume_List */
/*These statements are not active at the present time.  They are */
/*examples, and can be activated by removing the comment delimeters */
/*****************************************************************/
USER_VOLUME_LIST <NW1A25>
USER_VOLUME_LIST <NW1A26>
USER_VOLUME_LIST <NW1A27>
USER_VOLUME_LIST <NW1A28>
USER_VOLUME_LIST <NW1A29>
/*  User_Volume_List USRP01    */
/*  User_Volume_List USRP02    */
```

4. Save your changes with the **FILE** subcommand. Verify the integrity of the changes with the **CPSYNTAX** command then put the MAINT CF1 disk back online. The following example shows how you did this previously:

```
==> acc 193 g
==> cpsyntax system config f
CONFIGURATION FILE PROCESSING COMPLETE -- NO ERRORS ENCOUNTERED.
==> rel f (det
DASD 0CF1 DETACHED
==> cpacc * cf1 a
CPACCESS request for mode A scheduled.
HCPZAC6732I CPACCESS request for MAINT's 0CF1 in mode A completed.
==> q cpdisk
Label  Userid   Vdev Mode Stat Vol-ID Rdev Type   StartLoc     EndLoc
MNTCF1 MAINT    0CF1  A   R/O  520RES 0200 CKD          39         83
MNTCF2 MAINT    0CF2  B   R/O  520RES 0200 CKD          84        128
MNTCF3 MAINT    0CF3  C   R/O  520RES 0200 CKD         129        188
```

Verifying the changes

The system will see these changes the next time you IPL it. However, you need to make it see the changes now. You can do that by dynamically adding the PAGE DASD to the CP owned volumes:

```
define cpown slot 6 LX6PG2
Ready; T=0.01/0.01 13:58:38
```

After having accomplished it, have a look at the page space, as shown in Example 4-4, by using the **QUERY ALLOC PAGE command**. This is analogous to the **DISPLAY ASM** command in z/OS, and the Linux **free** command.

Example 4-4 Cheking out the newly created PAGE volume.

```
q alloc page
                EXTENT     EXTENT  TOTAL  PAGES   HIGH    %
VOLID  RDEV      START        END  PAGES IN USE   PAGE USED
------ ---- ---------- ---------- ------ ------ ------ ----
LX6PAG 1A22          1       3338 600840      0      0   0%
LX6PG2 8228          1       3338 601020     13     51   0%
                                  ------ ------        ----
SUMMARY                            1174K     13          1%
USABLE                             1174K     13          1%
Ready; T=0.01/0.01 14:40:42
```

To check out that the PERM volumes are now online, you should query the DASDs that are seen by the system, as shown in Example 4-5, by using the **QUERY DASD** command.

Example 4-5 Checking out the newly created PERM volumes

```
q dasd
DASD 1A20 CP OWNED  LX6RES   78
DASD 1A21 CP OWNED  LX6SPL   1
DASD 1A22 CP OWNED  LX6PAG   0
DASD 1A23 CP OWNED  LX6W01   113
DASD 1A24 CP OWNED  LX6W02   4
DASD 1A25 CP SYSTEM NW1A25   0
DASD 1A26 CP SYSTEM NW1A26   0
DASD 1A27 CP SYSTEM NW1A27   0
DASD 1A28 CP SYSTEM NW1A28   0
DASD 1A29 CP SYSTEM NW1A29   0
DASD 8228 CP OWNED  LX6PG2   0
Ready; T=0.01/0.01 14:48:20
```

Create the Linux Maintenance user

The following step to set up your environment before bringing Linux up is to create a user to host common files that will be used by every Linux guest system that you create. By doing so, all you need to do to bring as many Linux guests you want is to create a boot-parameter file for each of them instead of going through a painfull manual process everytime. In addition to that, the files needed for booting Linux won't be duplicated in every guest system user space.

You will have to be logged in as MAINT to perform the following steps. As stated in , "z/VM users and password planning" on page 145, the Linux administrator ID is LNXMAINT. Before making any changes to the USER DIRECT file to create its disks, make a copy of that file first:

```
==> copy user direct c = direorig = (oldd
```

Setting up minidisk space for LNXMAINT

A small 20 cylinder minidisk is allocated at virtual address *191* and a larger 300 cylinder minidisk (approximately 225MB), to be shared by many guests, is defined at virtual address *192*. You can use one of your brand new DASDs designated as PERM space in your worksheet (Table B-1 on page 409), or you can use any available space on LX6W02 (which will most likely have some free space if you have been following this book's installation guide all along) since you will only use 320 cylinders for the minidisks.

> **Note:** Cylinder 0 should always be reserved for the label therefore you should start minidisks at cylinder 1 if you choose to use one of the brand new DASDs.

1. Edit the `USER DIRECT` file and add the following user ID definition to the bottom of the file:

```
==> x user direct c
====> bottom
====> a 6
...
USER LNXMAINT LNXMAINT 64M 128M BEG                 1
 INCLUDE IBMDFLT                                    2
 LINK TCPMAINT 592 592 RR                           3
 MDISK 0191 3390 <2050> 0020 <LX6W02> MR READ WRITE MULTIPLE  4
 MDISK 0192 3390 <2070> 0300 <LX6W02> MR ALL WRITE MULTIPLE   5
*                                                           6
...
====> file
```

Note the following points for the numbers in black:

1 User ID `LNXMAINT`, same password, default size of storage (main memory) designated to this user is 64MB plus 128MB of extended storage, with class B, E and G privileges

2 Include the profile named `IBMDFLT` (which is defined earlier in the `USER DIRECT` file)

3 Link to the `TCPMAINT 592` disk read-only for access to **FTP** and other TCP/IP commands

4 Define a 191 minidisk of `size` 20 cylinders from `volume LX6W02`

5 Define 192 minidisk of `size 300` cylinders (approximately 225MB) from `volume LX6W02` with the special read password of **ALL** which allows read access from any user ID without a disk password

6 An empty comment line for better readability.

2. Whenever an `MDISK` statement is added or modified in the `USER DIRECT` file you should always check for overlapping cylinders and gaps (gaps will only leave empty disk space, however z/VM will allow you to *shoot yourself in the foot* by defining multiple minidisks over the same disk space). This is done with the **DISKMAP** command, as previously stated in , “Creating a disk” on page 140:

```
==> diskmap user
The minidisks with the END option specified in this directory will
not be included in the following DISKMAP file.

File USER DISKMAP A has been created.
```

3. The file created, `USER DISKMAP A`, contains a mapping of all minidisk volumes defined in the `USER DIRECT` file. It lists any overlaps or gaps found on the

volumes. Edit the file and turn off the prefix area with the **XEDIT PREFIX OFF** subcommand to view 80 columns:

```
==> x user diskmap
====> prefix off
```

4. Search for the text **overlap** with the / subcommand:

```
====> /overlap
```

You should see the error message: `DMSXDC546E Target not found`. This means that no minidisks are overlapping each other.

Look at the rest of the file. Your `LNXMAINT`'s minidisks definitions should look similar to ours, as depicted in Figure 4-17. Notice that it should be defined on the `LX6W02` disk, unless you are using one of your new DASDs.

```
00236 -----------------------------------------------------------
00237
00238 VOLUME USERID   CUU DEVTYPE      START END   SIZE
00239 LX6W02 $ALLOC$  A03 3390         00000 00000 00001
00240        40SASF40 2D2 3390         00001 00150 00150
00241        OSADMIN2 191 3390         00151 00160 00010
<...snip...>
00331        CERON    191 3390         02040 02049 00010
00332        LNXMAINT 191 3390         02050 02069 00020
00333        LNXMAINT 192 3390         02070 02369 00300
```

Figure 4-17 USER DISKMAP file after the creation of LNXMAINT.

5. Get out of the file `USER DISKMAP` with the **QUIT** command or by pressing **F3**.
6. If you have used one of your brand new DASDs, edit the `USER DIRECT` file again and add a new minidisk definition for that one very first cylinder you left out on it[8]. This prevents the system from reporting it as a gap.

```
==> x user direct
====> /user $alloc
USER $ALLOC$  NOLOG
 MDISK A01 3390 000 001 LX6RES R
 MDISK A02 3390 000 001 LX6W01 R
 MDISK A03 3390 000 001 LX6W02 R
 MDISK A04 3390 000 001 <NW1A25> R
```

7. Now that you are sure the minidisk layout is correct, the changes to the `USER DIRECT` file can be brought online via the **DIRECTXA** command:

[8] If you didn't leave it out, than you probably didn't read through the NOTE in the beginning of "Setting up minidisk space for LNXMAINT" on page 188.

```
==> directxa user
z/VM USER DIRECTORY CREATION PROGRAM - VERSION 5 RELEASE 3.0
EOJ DIRECTORY UPDATED AND ON LINE
HCPDIR494I User directory occupies 39 disk pages
```

If the **DIRECTXA** command fails, you must correct the problem before proceeding.

You have now defined your first z/VM user ID named LNXMAINT.

There is no equivalent process to this step in Linux because a user is not an entire virtual machine by itself, as it is in z/VM. When a Linux user is created, the information about it (user ID, password) is stored in a system file and a home folder is created for it in the /home directory.

Getting in and customizing the new user ID

Now you should be able to logon to the new user ID and format its two minidisks.

1. **Logoff** of MAINT and **logon** to LNXMAINT.

   ```
   LOGON LNXMAINT
   z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
   built on IBM Virtualization Technology
   There is no logmsg data
   FILES: 0003 RDR,   NO PRT,   NO PUN
   LOGON AT 05:41:34 EST THURSDAY 01/04/07
   z/VM V5.3.0    2007-11-10 10:07

   DMSACP112S A(191) device error
   ```

   You should see an error message ending in “device error”. When CMS is started, it tries to access the user’s *191* minidisk as file mode *A*. The 191 minidisk has been defined to this user ID, however, it has never been formatted for a CMS file system[9].

   > **Note:** DASDs used solely for holding minidisks don’t need to be formatted as PERM space because each minidisk on it will be formatted later on. In that case, its enough to format cylinder and put a label on it.

2. To format this disk for CMS use the **FORMAT** command. It requires a parameter specifying the file mode to access the disk as, mode **A** in the following example:

   ```
   ==> format 191 a
   ```

[9] Formtting a DASD as PERM storage is different than formatting a user minidisk for a CMS file system.

```
DMSFOR603R FORMAT will erase all files on disk A(191). Do you wish
to continue?
Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
lxm191
DMSFOR733I Formatting disk A
DMSFOR732I 20 cylinders formatted on A(191)
```

3. Format the larger 192 disk as the D minidisk which should take a minute or two:

```
==> format 192 d
DMSFOR603R FORMAT will erase all files on disk D(192). Do you wish
to continue?
Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
lxm192
DMSFOR733I Formatting disk D
DMSFOR732I 300 cylinders formatted on D(192)
```

You have now formatted the two minidisks and accessed them as file modes A and D.

Copying a PROFILE XEDIT

Copy the PROFILE XEDIT from the MAINT 191 disk so that XEDIT sessions have a common interface among user IDs.

1. Use the **VMLINK** command to both link to the disk read-only and to access it as the highest available file mode. The default read password is **read**:

```
==> vmlink maint 191
ENTER READ PASSWORD:
==> read
DMSVML2060I MAINT 191 linked as 0120 file mode Z
```

2. Copy the PROFILE XEDIT to your A disk:

```
==> copy profile xedit z = = a
```

Creating a PROFILE EXEC

Create a simple PROFILE EXEC that will be run each time this user ID is logged on.

1. Create the new file and add the following lines. REXX EXECs must always begin with a C language-style comment.

```
==> x profile exec a
====> a 5
```

```
/* PROFILE EXEC */
'acc 592 e'
'cp set run on'
'cp set pf11 retrieve forward'
'cp set pf12 retrieve'
====> file
```

This PROFILE EXEC access the TCPMAINT 592 disk as file mode E, sets CP run on (read section "Disconnecting" on page 64 for an explanation on the RUN parameter), and sets the retrieve keys per the convention.

There is some more nice configuration that you can put into your profile. We don't mention them here because they do not apply to the installation steps themselves, but you can have a look at them in 5.2, "Running the system" on page 230.

2. You could test your changes by logging off and logging back on. However, typing the command **PROFILE** will do the same. By default CMS tries to access the 191 disk as A and the 192 disk as D. Also you should have the TCPMAINT 592 disk accessed as E. To see your minidisks, use the **QUERY DISK** command:

```
==> profile
DMSACP723I E (592) R/O
==> q disk
LABEL  VDEV M  STAT CYL  TYPE  BLKSZ FILES BLKS USED-(%) BLKS LEFT BLK TOTAL
LXM191 191 A   R/W  i20  i3390 4096      2          9-01      3591      3600
LXM192 192 D   R/W  300  i3390 4096      0         11-00     53989     54000
TCM592 592 E   R/O  i67  i3390 4096    877       8167-68      3893     12060
MNT190 190 S   R/O  100  i3390 4096    689      14325-80      3675     18000
MNT19E 19E Y/S R/O  250  3390  4096   1010      26665-59     18335     45000
MNT191 120 Z   R/O  175  i3390 4096     36        224-01     31276     31500
```

3. Verify that your F11 and F12 keys are set to the **RETRIEVE** command:

```
==> q pf11
PF11 RETRIEVE FORWARD
==> q pf12
PF12 RETRIEVE BACKWARD
```

Getting the files needed for linux boot

In section "Setting up minidisk space for LNXMAINT" on page 188, you created two disks for the LNXMAINT user, the *191* and *192* disks. The 191 disk is to be used as LNXMAINT's "home" disk, where its generic files will be. On the other hand, the 192 disk will be used to host all of the files needed for a Linux virtual machine to boot. Later in "Setting up common profiles for the linux VMs" on page 198 you will set up a profile in the Linux guest that will link its 191 disk to LNXMAINT's 192 disk in read-only mode. By doing so, the guest's "home" disk will be able to load the necessary Linux files by reading them from this linked LNXMAINT disk.

Note: To proceed with this section, make sure you have set up an FTP server in your z/VM, as mentioned in , “Setting up an FTP server” on page 169. Again, sending the files through FTP is required to ensure that the transfer keeps the 80-character record length of the kernel and ramdisk files.

You have to send three files to your `LNXMAINT 192 disk:`

- the Linux kernel
  - In SLES10, this is the /boot/s390x/vmrdr.ikr file of its first DVD or CD
  - In RHEL5, this is the /images/kernel.img file of its first DVD or CD
- the Linux ramdisk
  - In SLES10, this is the /boot/s390x/initrd file of its first DVD or CD
  - In RHEL5, this is the /images/initrd.img file of its first DVD or CD
- the `SWAPGEN` file found in the compressed file pointed by this footnote[10].

Note: If your kernel and ramdisk are not in their 80-character record length, the kernel will die in a panic.

The authors were given the access information for a SLES10SP2 and a RHEL5 U1 repositories. Because we had to pick up one on which to base our examples here, we decided to go on with SLES. However, comments for a RHEL system are highlighted when a different step or configuration is needed.

Get the previous three files together into the same directory on your desktop (the machine you are using to connect to the mainframe) and use an FTP client to connect to the server, as shown in Example 4-6. Make sure you are logged off of `LNXMAINT` at this time or your ftp client won’t be able to access the disks in read-write mode.

Example 4-6 Send the files to LNXMAINT’s 192 disk via FTP

```
# ftp <9.12.4.89> 1
Connected to 9.12.4.89.
220-FTPSERVE IBM VM Level 530 at VMLINUX6.ITSO.IBM.COM, 16:33:09 EDT
THURSDAY 2008-05-15
220 Connection will close if idle for more than 5 minutes.
Name (9.12.4.89:root): lnxmaint 2a
331 Send password please.
Password:lnxmaint 2b
```

10 ftp://www.redbooks.ibm.com/redbooks/SG246695/

```
230 LNXMAINT logged in; working directory = LNXMAINT 191
Remote system type is z/VM.
ftp> cd lnxmaint.192 3
250 Working directory is LNXMAINT 192
ftp> put swapgen.exec 4
local: swapgen.exec remote: swapgen.exec
500 Unknown command, 'EPSV'
227 Data transfer will passively listen to 9,12,4,89,4,7
125 Storing file 'swapgen.exec'
100% |*************************************| 16750      14.60 MB/s
--:-- ETA
250 Transfer completed successfully.
16750 bytes sent in 00:00 (1.74 MB/s)
ftp> bin 5
200 Representation type is IMAGE.
ftp> site fix 80 6
200 Site command was accepted.
ftp> put vmrdr.ikr sles10s2.ikr 7
local: vmrdr.ikr remote: sles10s2.ikr
227 Data transfer will passively listen to 9,12,4,89,4,8
125 Storing file 'sles10s2.ikr'
100% |*************************************|  5904 KB    4.48 MB/s
00:00 ETA
250 Transfer completed successfully.
6046280 bytes sent in 00:01 (4.39 MB/s)
ftp> put initrd sles10s2.initrd 8
local: initrd remote: sles10s2.initrd
227 Data transfer will passively listen to 9,12,4,89,4,9
125 Storing file 'sles10s2.initrd'
100% |*************************************|  8067 KB    3.32 MB/s
00:00 ETA
250 Transfer completed successfully.
8261468 bytes sent in 00:02 (3.28 MB/s)
ftp> quit 9
221 Quit command received. Goodbye.
```

Note the following remarks to understand what you just did for each of the highlighted steps:

1	ftp to the z/VM server
2a	Log in as LNXMAINT
2b	Type in LNXMAINT's password
3	Change the working "directory" on the server to LNXMAINT's 192 disk

4	Send the swapgen.exec file as a text file
5	Change ftp mode to binary
6	Set the record format to fixed 80-byte records
7	Send the kernel image and rename it to sles10s2.ikr on the server
8	Send the ramdisk image and rename it to sles10s2.initrd on the server
9	quit

Now check that your files have been properly uploaded. Log in as `LNXMAINT` and list your *D* file mode. You should see a listing similar to the one in Figure 4-18. Make sure that your *SLES10S2 IKR* and *SLES10S2 INITRD* files have a record length of 80 bytes.

```
 LNXMAINT FILELIST A0  V 169  Trunc=169 Size=13 Line=1 Col=1 Alt=0
Cmd   Filename Filetype Fm Format Lrecl Records     Blocks   Date Time
      SLES10S2 INITRD   D1 F         80 103269        2017  5/15/08 16:34:07
      SLES10S2 IKR      D1 F         80 75579         1122  5/15/08 16:33:59
      SWAPGEN  EXEC     D1 V         72 358              5  5/15/08 16:33:21

1= Help       2= Refresh  3= Quit   4= Sort(type)  5= Sort(date)  6=
Sort(size)
7= Backward   8= Forward  9= FL /n 10=          11= XEDIT/LIST 12= Cursor

====>
                                                          X E D I T  1 File
```

Figure 4-18 Checking the uploaded kernel, ramdisk and swapgen files.

The Linux kernel loader script and boot parameters file

In order to have your Linux guest machines load the Linux kernel you just uploaded to `LNXMAINT`, you should create a script to automate the whole process. Also, it will be available to every Linux guest you create because you will place it on `LNXMAINT`'s *D* file mode. Do this by issuing the command **`XEDIT SLES10S2 EXEC D`** and paste the contents of Example 4-7 into it.

> **Note:** REXX executables **must** begin with a C-like comment, **/*   */** as shown in the example below.

Example 4-7 Script to load the Linux kernel

```
/* EXEC punches sles10 sp2 kernel/ramdisk to reader and ipls it */
```

```
'cp spool pun *'
'cp close rdr'
'pur rdr all'
'pun sles10s2 ikr * (NOH'
'pun' userid() 'parmfile * (NOH' 1
'pun sles10s2  initrd * (NOH'
'ch rdr all keep'
'ipl 00c clear'
```

You are only one step away from having all of the necessary files ready for a Linux boot. In fact, you are missing the file denoted number **1** that is used by the loader script of Example 4-7 on page 196.

In order to save you some time and typing, it is wise to create a file with all of the boot parameters you will need to perform a network-based installation of SLES10. Parameters such as the IP address to be used by the Linux guest, the netmask, the server repository URL, and so on can be placed into a single file and read by YaST[11] at boot time. Because this information is particular to each Linux guest that you create, you should name this file as <LINUX_GUEST> PARMFILE D, where the string <LINUX_GUEST> is the name of your z/VM user that will host the Linux system. Notice that the instruction marked with **1** in Example 4-7 on page 196 makes use of the `userid()` REXX command to come up with the complete file name. This means that each user will punch a different parameters file when they run this script from within their Linux guest environments.

For now, you don't have a user for the Linux guest because you will only create one in section , "`Create the first Linux VM" on page 199`. So, right now all you should do is to name this parameter file after the name you intend to give to your Linux guest user. In our examples, we will use `LNXCER` for the Linux guest.

Example 4-8 Boot parameter file for `Linux guest LNXCER`

```
ramdisk_size=65536 root=/dev/ram1 ro init=/linuxrc TERM=dumb
hostip=<9.12.5.65> hostname=<ceron>
gateway=<9.12.4.1> netmask=<255.255.254.0>
broadcast=<9.12.7.255> layer2=0
readchannel=0.0.0600 writechannel=0.0.0601 datachannel=0.0.0602
nameserver=<9.12.6.7> portname=dontcare
install=<ftp://totibm:itso@9.12.4.69/code/sles10x>
instnetdev=osa osainterface=qdio osamedium=eth Manual=0
usessh=1 sshpassword=letmein
```

[11] YaST stands for *Yet Another Setup Tool*, and is the installer for SLES.

Make sure that you replace the <value> entries with the ones that correspond to your resources according to your planning worksheet at "Planning worksheet" on page 409. Additionally, don't forget the last line highlighted in bold since in this book we instruct you to perform an SSH installation of SLES10.

Setting up common profiles for the linux VMs

In order to have all Linux guests see the *192* disk you created for LNXMAINT with all of the common files for boot, you have basically two options:

- Put a **LINK** statement for every user description in MAINT's USER DIRECT file, which is the stupid way of doing this, or
- Create a profile based on the default profile and add the proper lines for the link statements, so that all you have to do when creating a Linux guest user to make use of those is to import this new profile

Our examples are based on the second approach, of course.

To accomplish this, perform the following steps:

1. Logon to MAINT and edit the USER DIRECT file:

   ```
   ==> x user direct c
   ```

   In the USER DIRECT file you can group statements that will be common to many user definitions in a construct called a *profile*. This profile can then become part of user definitions via the INCLUDE statement. You used the existing profile TCPCMSU when you defined the LNXMAINT user.

2. Create a new profile named LNXDFLT. This will contain the user directory statements that will be common to all Linux user IDs. To save typing, you can use the "" prefix commands to duplicate the IBMDFLT profile that should be on lines 37-50:

   ```
   ""037
   ****************************************************************
   00038 *
   00039 PROFILE IBMDFLT
   00040   SPOOL 000C 2540 READER *
   00041   SPOOL 000D 2540 PUNCH A
   00042   SPOOL 000E 1403 A
   00043   CONSOLE 009 3215 T
   00044   LINK MAINT 0190 0190 RR
   00045   LINK MAINT 019D 019D RR
   00046   LINK MAINT 019E 019E RR
   00047   LINK MAINT 0402 0402 RR
   00048   LINK MAINT 0401 0401 RR
   00049   LINK MAINT 0405 0405 RR
   ""050 ******************************
   ```

3. Edit the duplicated profile by deleting the three `LINK MAINT 040x` lines, and inserting the lines that are in bold text below:

```
PROFILE LNXDFLT
  IPL CMS                                1
  MACHINE ESA 4                          2
  CPU 00 BASE                            3
  CPU 01                                 4
  NICDEF 600 TYPE QDIO LAN SYSTEM VSW1   5
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  CONSOLE 009 3215 T
  LINK MAINT 0190 0190 RR
  LINK MAINT 019D 019D RR
  LINK MAINT 019E 019E RR
  LINK LNXMAINT 192 191 RR               6
  LINK TCPMAINT 592 592 RR               7
```

Notes:

1 CMS will be IPLed when the user ID is logged onto
2 Machine will of type ESA with a maximum of 4 CPUs that can be defined
3 Defines the base CPU
4 Defines a second CPU - **don't include this** if your LPAR has only a single Real IFL/CP
5 Defines a virtual NIC connected to the VSWITCH starting at virtual address 600
6 Provides read access to the `LNXMAINT 192` disk as the user's 191 disk
7 Provides read access to the `TCPMAINT 592` disk, so that the user has access to TCPIP services such as an FTP client.

Save the file:
====> **file**

Create the first Linux VM

If you well remember the virtualization discussion we had at 2.2.1, "History of z/VM" on page 18, all you have to do to create a virtual machine for Linux to run into is to create a z/VM user. This is what you will accomplish in this section.

Setting minidisks up

Create a new user for your Linux guest and set up its minidisks exactly in the same way you did for the `LNXMAINT` user in "Setting up minidisk space for LNXMAINT" on page 188. Log in as `MAINT` and add a new user definition into the `USER DIRECT C` file.This will be your first user to include the **lnxdflt** profile you created in , “Setting up common profiles for the linux VMs” on page 198. Example 4-9 shows a template you can use for that.

Example 4-9 Create the Linux guest user.

```
user <lnxcer> <lnxcer> 512M 2G BDEG
include lnxdflt
option lnknopas applmon
mdisk 100 3390 0001 3338 <nw1a25> mr lnx4vm lnx4vm lnx4vm
mdisk 103 3390 0001 3338 <nw1a26> mr lnx4vm lnx4vm lnx4vm
mdisk 104 3390 0001 3338 <nw1a27> mr lnx4vm lnx4vm lnx4vm
mdisk 105 3390 0001 3338 <nw1a28> mr lnx4vm lnx4vm lnx4vm
mdisk 106 3390 0001 3338 <nw1a29> mr lnx4vm lnx4vm lnx4vm
```

This Linux user ID will have the following minidisks and virtual disks:

100 This minidisk will hold the /boot partition and have the remaining space given to the system LVM, where all of the other partitions will be in. More about Linux LVM is described in “Linux Logical Volume Manager” on page 308.

101-102 These are virtual disk (VDISK) swap spaces created by **SWAPGEN EXEC** upon logging on. They are NOT defined in USER DIRECT, but dynamically in the user’s **PROFILE EXEC** when the user ID logs on.

103-106 These minidisk will be managed by the Linux system LVM.

> **Important:** A minimalistic Linux guest system fits onto a single 3390-3 DASD, and this is the recommended practice in the field. Such practice requires the use of no GNOME or KDE window managers to keep the installed system small. Our example does not do this because we want to show the use of LVM and KDE.

To avoid a 1-cylinder gap being reported on each user volume, it is recommended to use the user ID $ALLOC$. This user is set to NOLOG which means it can never be logged onto. Thus, it is not a conventional user ID, rather it is a convenient place to put dummy minidisks definitions for cylinder 0 of all PERM volumes. You will find ID $ALLOC$ in the beggining of the USER DIRECT C around line 100. Include a definition for each of your volumes, as shown in Example 4-10.

Example 4-10 Avoiding a 1-cylinder gap being reported

```
00104 *
00105 USER $ALLOC$  NOLOG
00106  MDISK A01 3390 000 001 LX6RES R
00107  MDISK A02 3390 000 001 LX6W01 R
```

```
00108  MDISK A03 3390 000 001 LX6W02 R
00109  MDISK A04 3390 000 001 <NW1A25> R
00110  MDISK A05 3390 000 001 <NW1A26> R
00111  MDISK A06 3390 000 001 <NW1A27> R
00112  MDISK A07 3390 000 001 <NW1A28> R
00113  MDISK A08 3390 000 001 <NW1A29> R
00114 *
```

Update the autolog profile

The new Linux ID you defined needs access to the VSWITCH. A **SET VSWITCH** command with the **GRANT** parameter can be added to AUTOLOG1's **PROFILE EXEC** to do this automatically when the system IPLs. Also, an **XAUTOLOG** statement can be added for your Linux user ID to be automatically logged on at z/VM IPL time.

To accomplish those, log in as MAINT and make sure your AUTOLOG1's profile looks like the one shown in Example 4-11.

Example 4-11 Granting users access to the virtual switches

```
==> link autolog1 191 1191 mr
==> acc 1191 f
==> x profile exec f   // add two lines
/***************************/
/*  Autolog1 Profile Exec  */
/***************************/
'cp xautolog tcpip'              /* start up TCPIP */
'CP XAUTOLOG DTCVSW1'            /* start VSWITCH controller 1 */
'CP XAUTOLOG DTCVSW2'            /* start VSWITCH controller 2 */
'cp set pf12 ret'                /* set the retrieve key */
'cp set mdc stor 0m 128m'        /* Limit minidisk cache in CSTOR */
'cp set mdc xstore 0m 0m'        /* Disable minidisk cache in XSTOR */
'cp set srm storbuf 300% 250% 200%' /* Overcommit memory */
'cp set signal shutdown 180'     /* Allow guests 3 min to shut down */

/* Grant access to VSWITCH for each Linux user */
'cp set vswitch vsw1 grant <lnxcer>'

/* XAUTOLOG each Linux user that should be started */
'cp xautolog <lnxcer>'

'cp logoff'                       /* logoff when done */
====> file
```

These changes will not take effect until the next IPL, so you must grant this user ID access to the VSWITCH for this z/VM session. This is done by running the following command, as the MAINT user:

```
==> set vswitch vsw1 grant <lnxcer>
Command complete
```

Set up the Linux guest profile

This is the last step before you run Linux in your guest system by the first time. This profile should be common among the guests you create, so the best place to put it is into LNXMAINT's 192 disk, which will be mapped as the A file mode of those guests by the lnxdftl profile that you created in "Setting up common profiles for the linux VMs" on page 198.

Log in as LNXMAINT and and make the PROFILE EXEC D file look like as shown in Example 4-12.

Example 4-12 Linux guests profile

```
/* PROFILE EXEC */
 'CP SET RUN ON'
 'CP SET PF11 RETRIEVE FORWARD'
 'CP SET PF12 RETRIEVE'
 'ACC 592 C'
 'SWAPGEN 101 524288'                                    1
 'SWAPGEN 102 1048576'                                   2
 'PIPE CP QUERY' userid() '| var user'
 parse value user with id . dsc .
 ipldisk=100
 if (dsc = 'DSC') then                                   3
 'CP IPL' ipldisk
 else                                                    4
 do
 say 'Want to ipl Linux from DASD' ipldisk'? y/n'
 parse upper pull answer .
 if (answer = 'Y') then
 'CP IPL' ipldisk
 end
```

This profile has some interesting aspects, as denoted by the numbers to the right of some of its lines:

1,**2** These lines will create a swap space as disks *101* and *102* for Linux each time the virtual machine is started.

Note: It's recommended that you define the minimum amount of swap space as possible, depending on your guest system's purposes. Doing so optimizes your z/VM system storage management. In our example, we used 512MB and 1GB because we were not worried about performance nor memory contraints.

3 This statement uses the parsed value stored in the `DSC` variable to check out whether or not this user is being autologged in. If it is the case, then the script will IPL what is in disk *100*, which is where the Linux image should be after the installation.

4 In case the previous check concludes that the user actually performed an ordinary login, then the script will ask you if you want to IPL Linux on disk *100*.

Let's now install Linux onto our guest. Log in as your Linux guest user and press Enter to run its profile. You should see an output like the one in Figure 4-19.

```
00: ipl cms
z/VM V5.3.0    2008-05-06 13:40

DMSACP723I A (191) R/O
DMSACP723I C (592) R/O
DIAG swap disk defined at virtual address 101 (64989 4K pages of
swap space)
DIAG swap disk defined at virtual address 102 (129981 4K pages of
swap space)
Want to ipl Linux from DASD 100? y/n
n
Ready; T=0.01/0.04 09:49:42
```

Figure 4-19 Logging in as the Linux guest user

Notice that the swap space will be automatically generated as we mentioned earlier. Also notice that, since this was not an auto login, the system prompted you on whether to IPL disk 100 or not. At this point, answer **n**, for no, because you don't have any Linux image installed onto it yet.

4.4.2 Booting the Linux kernel for installation

Because you invested your time in configuring a common Linux user ID (`LNXMAINT`) before, now all you have to do to start the Linux install is to call the

SLES10S2 EXEC script that you created in "The Linux kernel loader script and boot parameters file" on page 196. Figure 4-20 on page 205 shows how to call it and the resulting output until the Linux kernel starts running.

```
Ready; T=0.01/0.04 09:49:42
sles10s2
00: 0000003 FILES PURGED
00: RDR FILE 0043 SENT FROM LNXCER PUN WAS 0043 RECS 076K CPY  001 A NOHOLD
NOKEEP
00: RDR FILE 0044 SENT FROM LNXCER PUN WAS 0044 RECS 0009 CPY  001 A NOHOLD
NOKEEP
00: RDR FILE 0045 SENT FROM LNXCER PUN WAS 0045 RECS 103K CPY  001 A NOHOLD
NOKEEP
00: 0000003 FILES CHANGED
00: 0000003 FILES CHANGED
Linux version 2.6.16.21-0.8-default (geeko@buildhost) (gcc version 4.1.0
(SUSE L
inux)) #1 SMP Mon Jul 3 18:25:39 UTC 2006
We are running under VM (64 bit mode)
Detected 2 CPU's
Boot cpu address  0
Built 1 zonelists
Kernel command line: ramdisk_size=65536 root=/dev/ram1 ro init=/linuxrc
TERM=dumb hostip=9.12.5.65 hostname=ceron
          gateway=9.12.4.1 netmask=255.255.254.0
          broadcast=9.12.7.255 layer2=0
          readchannel=0.0.0600 writechannel=0.0.0601 datachannel=0.0.0602
          nameserver=9.12.6.7 portname=dontcare
          install=ftp://totibm:itso@9.12.4.69/code/sles10x
          instnetdev=osa osainterface=qdio osamedium=eth Manual=0
          usessh=1 sshpassword=letmein
PID hash table entries: 4096 (order: 12, 131072 bytes)
Dentry cache hash table entries: 131072 (order: 8, 1048576 bytes)
Inode-cache hash table entries: 65536 (order: 7, 524288 bytes)
Memory: 496640k/524288k available (4298k kernel code, 0k reserved, 1401k
data, 1
96k init)
Security Framework v1.0.0 initialized
Mount-cache hash table entries: 256
checking if image is initramfs... it is
Freeing initrd memory: 8067k freed
cpu 0 phys_idx=0 vers=FF ident=04DE50 machine=2097 unused=8000
cpu 1 phys_idx=1 vers=FF ident=04DE50 machine=2097 unused=8000
Brought up 2 CPUs
migration_cost=1000
NET: Registered protocol family 16
debug: Initialization complete
TC classifier action (bugs to netdev@vger.kernel.org cc hadi@cyberus.ca)
```

Figure 4-20 Booting the Linux kernel for installation.

At this point, your Linux kernel is up and running and the ramdisk is taking care of downloading the installation stage 2 file from your server repository. This stage 2 file is actually the one that contains the installer program itself, YaST in case of SLES10. Since we made the option of doing the installation via SSH, the boot should end up by displaying a message similar to the one in Figure 4-21.

```
setting root pwd to 'letmein'
Starting SSH daemon  ...

eth0: <BROADCAST,MULTICAST,UP> mtu 1492 qdisc pfifo_fast qlen 1000
    link/ether 02:00:01:00:00:02 brd ff:ff:ff:ff:ff:ff
    inet 9.12.5.65/23 brd 9.12.5.255 scope global eth0
    inet6 fe80::200:100:400:2/64 scope link
       valid_lft forever preferred_lft forever

      ***  sshd has been started  ***

      ***  login using 'ssh -X root@ceron'  ***
      ***  run 'yast' to start the installation  ***
```

Figure 4-21 YaST ready for an SSH installation

The reason why we chose SSH over the IBM Connection terminal is actually that an SSH connection can refresh the screens during the install wizard and its a colorful console.

Note: Read *z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES9*, SG24-6695 for a graphical installation of SLES10 by using a VNC client.

Author Comment: change the pointer above to sles10 cookbook later <ceron>

To proceed with the installation, open an SSH connection to the Linux guest (you can do this with the Linux **ssh** command or use an SSH client, such as **putty**, if you are in Windows. Login in as `root` and use the `letmein` password defined to protect the root user during the installation. After you are logged in, type **yast** to start the installation.

Note: You will be prompted to set the real root password of your system later on during the installation. Don't worry about having a weak root password while installing the system.

Beginning an YaST installation

When YaST comes online onto your SSH connection, it will guide you through an installation wizard. Perform the following steps:

Hint: The `Tab` key will cycle you forward in the menu entries. The `Escape+Tab` combination will cycle you backward in the menu entries. `Space` will toggle a checkbox or open up a selection box.

The installation program that is running is **`yast2`**. Perform the following steps:

1. Choose the language, **English**, (or your language) and click Next.
2. The *SLES10 SP2 license agreement window* should appear. Choose **Yes, I agree to the License Agreement** and Click Next.
3. The *Disk Activation* window should appears. Choose **Configure DASD Disks**.
4. The *YaST2 DASD Disk Management* window should appear: you will see all the DASD available to your Linux guest, also including the z/VM disks that your guest user uses for itself and not for hosting Linux, such as its 191 disk.
   a. Use only the 100-106 disks for your Linux install because that is what you have set up for it.
   b. **Highlight each of them** and click **Select or Deselect** to select them, or simply press **`Enter`** while highlighting a given disk.
   c. You should see a the word *Yes* next to the disks you just selected. Activate them for the Linux you're about to install. Click **Perform Action** -> **Activate**.
   d. The disks that are not meant for swapping must be formatted so that Linux can use them. Deselect disks 101 and 102, so that the other ones for the Linux image remain selected. Now click **Perform Action** -> **Format** .
5. You should see a window asking for one Parallel Formatted Disks: click **OK**.
6. Click **Yes** to the question *Really format the following?*
7. A progress indicator window should appear displaying progress. This step can take 5-15 minutes depending on the type of channel and the speed of the disks.

8. When the formatting is complete, click **Next**, in *Disk Activation* window click **Next** again.
9. A window will appear asking for the installation mode: select **New installation** and Click **Next**
10. The *Clock and Time Zone* window will appear. Choose your time zone and set clock, click **Next**.
11. This will bring you to the *Installation Settings* window. Click **Partitioning**. The *Expert Partitioner* window will appear .

    We use LVM to join our 5 DASDs together and make themlook like one single 10GB disk to Linux. By doing so, we can make Linux partitions, such as /home, to spawn multiple disks and grow to as much as we want. However, there is one partition that needs to be left out of the LVM in order for the system to be able to boot: the */boot* partition.

> **Note:** Refer to the following articles for a discussion on LVM (we recommend you really go through this):
>
> - http://www.ibm.com/developerworks/linux/linux390/perf/tuning_rec_dasd_optimizedisk.html
> - http://www.ibm.com/developerworks/linux/linux390/perf/tuning_rec_dasd_volMan.html
>
> **Important:** A minimalistic Linux guest system fits onto a single 3390-3 DASD, and this is the recommended practice in the field. Such practice requires the use of no GNOME or KDE window managers to keep the installed system small. Our example does not do this because we want to show the use of LVM and KDE.

12. Click **Create** and because **/dev/dasda** has no partitions, you will be brought to *Create a Primary Partition on /dev/dasda*.
13. You should see a window similar to that shown.
    a. Click the **Format** radio button.
    b. Choose **Ext3** as the file system

> **Note:** The authors recommend that you do not use reiserfs because the file structure tree gets corrupted from time to time, which would force you to run an `fsck --rebuild-tree,` a time consuming and dangerous task. For a discussion on filesystem types, refer to http://www.ibm.com/developerworks/linux/linux390/perf/tuning_res_journaling.html

   c. Set the first cylinder to be *0* (zero). For the end cylinder, use `+64M` to denote that you want a partition that is 64MB in size. The `/boot` will only host the kernel, ramdisk and bootloader files, so this is more than enough space for it. 64MB is enough room to host about 4 to 5 different kernel and ramdisk images.

   d. Select a mount point of **/boot**.

   e. Click **OK**.

14. Now you should create another partition on `dasda` repeating the same actions described in steps *12* and *13*. The start cylinder will be displayed as the next available cylinder. Make sure the ending cylinder is the last one available (3337 for a 3390-3 DASD) and choose the empty mount point for it (the last choice in the mount point selection box).

15. Now, repeat step *14* for each of the other disks, except `dasdb` and `dasdc` because they are swap space, and make sure that the partition you create uses all of those disks space. You may safely ignore warnings that request you to create a root partition (/) for now. The goal of this step is only to make the system able to see a *Linux Native* partition on each of the other disks, otherwise LVM will fail to use them.

16. Back to the *Expert Partitioner* window, click **LVM** to proceed and create the LVM.

17. At this point, you should choose a volume group name for the LVM and a physical extend size. We recommend you use the defaults that YaST presents you. Click **`OK`** to proceed.

18. All of your remaining partitions will be shown in a list. Add each of them to the LVM by highlighting it and pressing **`Enter`** or by clicking on **`Add Volume`**. Click **`Next`** to proceed.

19. You should see a screen showing the `/boot` partition you created and the two swap partitions. This screen is the LVM Logical Volumes management screen. In the previous step you created a *logical group*, which can be understood as creating a logical disk, or DASD in a z/VM nomenclature. You now need to create the *logical volumes* inside the logical group. Logical volumes are nothing else than Linux partitions, such as */*, */home*, */var* and so on. Think of logical volumes as z/VM minidisks. Click **Add** .

20. Next perform the following steps:

   a. Format it as ext3.

   b. Give it a meaningfull logical volume name. Since this will be our root partition, we named it as *root.*

   c. Set its size. We are using 8GB here because we intend to create a /home partition around 3GB later.

d. Use the default value for *stripes*, unless you want to optimize the system for performance, as stated earlier in the LVM reference we provided.

e. Pick up / as the mount point

f. Click **OK**

21. Repeat step 20 and create a */home* partition. Name its volume as *home* and click the *[max]* button when setting up the space for it in order to use all of the remaining space in the volume group. Click **Next** to proceed to the main partitioning menu and then **Finish** to get out of it.

Continuing the YaST installation

You should be back to the main wizard right at the point before you started step 11 of the previous section. All you have to do here now before starting to install packages is selecting which packages you want to install. Follow these steps to accomplish that:

1. Click **Change** → **Software**
2. Filter the packages by patterns by clicking the **Filter** selection box.
3. Select any patterns you want for install. Since we will give examples on how to connect to a KDE environment through a VNC connection, we will deselect the GNOME Desktop environment and select the KDE one. Click **OK** when you are done.
4. At this point, you will see a list of the packages scheduled for installation, based on the patterns you chose in the previous step. Click **Accept** to proceed back to the install wizard. You may need to accept some dependencies depending on the patterns selected. If that happens, click **OK** to continue.
5. You are back to the main wizard menu. It's now time to click **Accept** to start the installation. A confirmation screen will be displayed.
6. Click **Install** and watch the installation take place. You will be looking at a packates installation screen for a short while.

Booting your new Linux system from disk

After the first part of installation completes, your Linux system will shutdown and you will need to return to your z/VM 3270 session. **IPL** the newly installed system from disk to continue installation. Issue the following command and your new kernel should boot from disk:

```
Power down.
00: HCPGSP2629I The virtual machine is placed in CP mode due to a
SIGP stop from
 CPU 01.
```

```
01: HCPGSP2630I The virtual machine is placed in CP mode due to a
SIGP stop and
store status from CPU 01.
```

Enter **IPL 100** and Linux will be IPLed.

```
==> ipl 100
00: zIPL v1.6.3 interactive boot menu
00:
00:  0. default (ipl)
00:
00:  1. ipl
00:  2. failsafe
00:  3. ipl
00:  4. failsafe
00:
00: Note: VM users please use '#cp vi vmsg <number>
<kernel-parameters>'
00:
00: Please choose (default will boot in 10 seconds):
```

At this prompt, you can wait 10 seconds for the default IPL to proceed, or you can try to type **#cp vi vmsg 0** if you can type quickly enough.

The install program will bring up an SSH server again to complete the installation, as shown in Figure 4-22.

```
***  sshd has been started  ***

you can login now and proceed with the installation
run the command '/usr/lib/YaST2/startup/YaST2.ssh'

active interfaces:

eth0      Link encap:Ethernet  HWaddr 02:00:01:00:00:02
          inet addr:9.12.5.65  Bcast:9.12.5.255  Mask:255.255.254.0
--
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
```

Figure 4-22 YaST messages in first boot.

Completing YaST installation

Go back to the same SSH client used for the first part of installation.

Log in using the same SSH password (`letmein` in this example) as the one used to first bring up YaST.

1. You will now get a window for setting the root password. Enter your desired root password twice and click **Next**. *Don't forget* this password!!
2. The first window you will probably see is similar to the one you saw last time - package installation. However there is a good chance it will be done already, depending on your patterns selection, and you will come to *Hostname and Domain Name* window.
3. Enter the required values, in this example **ceron** is the host name and **itso.ibm.com** is the domain name. Uncheck the **Change Hostname via DHCP** checkbox, unless of course you have one DHCP server in your environment. Click **Next**.
4. In the *Network Configuration* window you will see *Firewall is enabled*, **click on the word enabled** to change it to *disabled*. Enable the VNC remote administration by clicking **`Change`** → **`VNC Remote Administration`**. All other values should be correct, so just click **Next**.

   > **Warning:** Disabling the firewall is not a good practice. We are doing so in order to simplify our examples here. You should later activate the firewall and block the ports for services you don't want your system to serve incoming connections at.

5. In the *Test Internet Connection* window select **No, skip this test** and click **Next**.
6. The next window will be *Installation Settings*. Select the **Skip configuration** radio button and click **Next**.
7. In the *User Authentication Method* window accept the default of **Local (/etc/passwd)** and click **Next**.
8. The next window will be *Add a new local user*. You may choose to add a local user if you don't want to use the system as `root` every time. When you are done, click **Next**.
9. In the *Writing the system configuration* window the **`SuSEconfig`** tool writes all your settings to disk.
10. The next window will be *Release Notes*. After reviewing the release notes, click **Next**.
11. In the *Hardware Configuration* window choose the **Skip Configuration** radio button and click **Next**.

12. The last installation window is *Installation Completed*. **Check** the check box *Clone This System for Autoyast* and click **Finish.** If you don't intend to clone the system, you may leave the check box unselected.

Congratulations! You are done installing Linux!

4.4.3 Logging into the Linux system

To close the installation guide in a great style, nothing better than logging into the system you have just installed in a nice and pleasant way: VNC. In order to do that, you must have a VNC client installed onto your machine. You can use Linux's **`vncviewer`** client or get one for Windows, such as RealVNC. In our examples we use RealVNC.

Connect to the guest via VNC

Start the RealVNC application and type in the IP address of the "server" (your Linux system) followed by the port number the VNC server is running on it. Figure 4-23 depicts our VNC client connecting to our Linux guest system.

Figure 4-23 Connecting to the Linux system via VNC

> **Note:** When you install SLES10 and enable VNC management, a VNC server will be running on port 5901 after the system boots up.

You should enter the the VNC server information according to the *IP:port* format shown in Figure 4-23.

RealVNC then brings you a new window displaying the X Window System. At this point, it is most probable that you won't see the KDM login manager, but instead you will see the X Window System.

In order to bring the login manager up, log in as `root` to the Linux guest via SSH and type **`/etc/init.d/xdm start`**. It may be a while for KDM to be displayed in your current VNC display depending on your network speed. You should now see a KDM login display, such as the one in Figure 4-24.

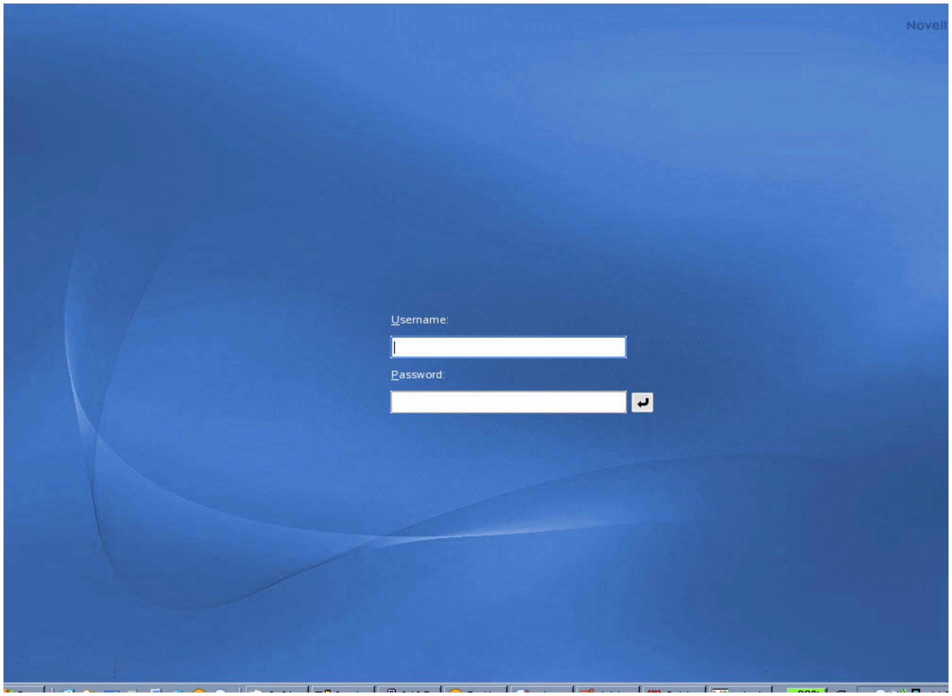

Figure 4-24 KDM Login screen

Log in as `root` and have a first look at your graphical KDE interface.

Configure KDM to autostart in boot

It is not desirable to have to start the XDM service everytime you want to log into the system with VNC. Therefore, you should enable it as a system service in the runlevels. Follow the steps denoted in Figure 4-25.

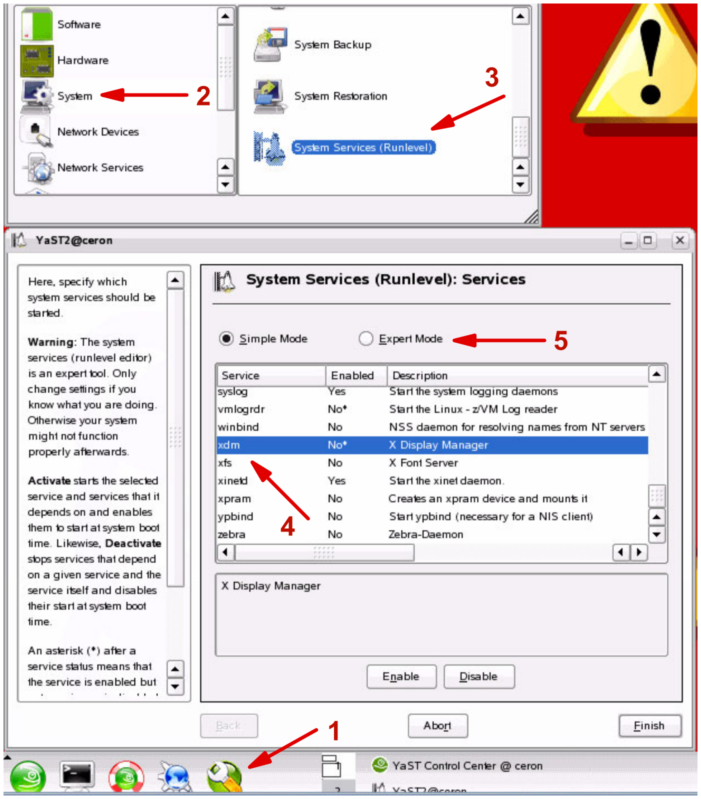

Figure 4-25 Enabling XDM as a system service.

1. Click the *YaST* icon
2. Scroll down and select *System* in the left menu
3. Once the right menu refreshes, scroll down and select *System Services (Runlevel)*
4. Scroll down the list until you see an entry for **xdm**
5. Highlight the xdm line and click **Expert Mode**
6. The window will change to an export mode as shown in Figure 4-26 on page 216.

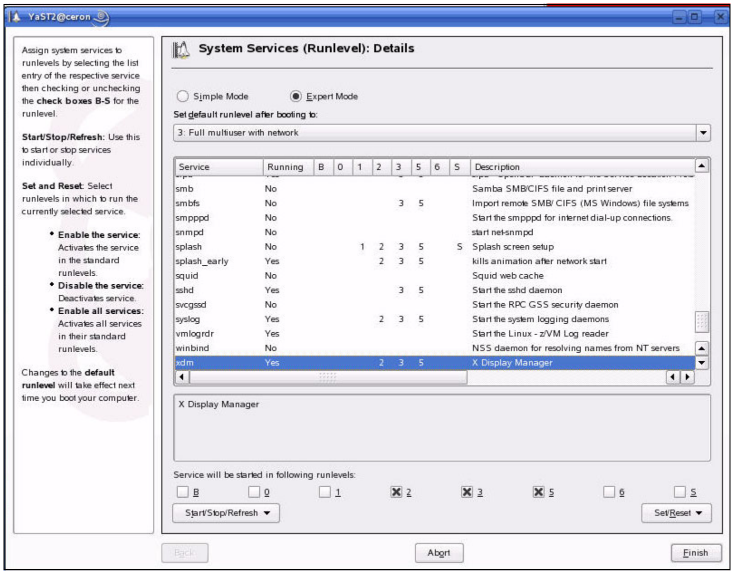

Figure 4-26 Runlevel editing in expert mode.

7. Enable XDM to be run on levels 2, 3 and 5. Click **`Finish`** to complete the process.

These steps should do the trick. The next time and on when you log in via a VNC client, you will see a KDM login screen.

For a deep dive into the peculiarities of a Linux installation, which we don't mention in this book, have a look at the online documentation in http://www.gentoo.org/doc/en/handbook/index.xml [12].

[12] Although this is the handbook of an specific Linux distribution, the concepts behind a Linux installation are very well outlined and generic.

4.5 Running a z/OS image as a z/VM guest

The process of running a z/OS system as a guest of z/VM is much the same as running any other guest system. In this section, we discuss moving a z/OS image from a z10 LPAR to run as a z/VM guest operating system. This should work for systems any System Z processor. This could be the scenario for many sites either due to LPAR constraints, server consolidation, or to simplify their test environment. One of the strengths concerning z/VM is its ability to virtualize both hardware in form of (for instance) virtualized Coupling Facilities, Channel-to-channel adapters and more. One could also manipulate the device addresses, thus minimizing the need for administration of the hardware configuration (IOCDS). The latter enables a quick and efficient path to set up new test images when desired.

This process is essentially the same as if a new z/OS system were being created to run as a z/VM guest. Determining the resources needed would still have to be planned and at the point were we IPL the existing z/OS system as a z/VM guest, the z/OS installation process would then be performed.

4.5.1 z/OS candidate description

The z/OS image we are moving from an LPAR to z/VM guest had specific resources defined to it. It had a SYSNAME of *SC59* and you may want use that as your z/VM userid, but it is not required. It had no CTCs and was not utilizing CFs or other HW to take into account besides communication via OSA adapter. It had a hardware configuration like listed in Table 2(simplified listing):

Type of resource	Amount/ description	Comments
Sysname	SC59	Can be the z/VM userid
Memory / CS	2 GB	Initial storage
Processors / CPs	4	
DASD / Disks	53 volumes	3390 Mod 3 and Mod 9
OSA network adapter	2000-2005	QDIO
Console	Address F008	3270 console

Table 2 Table of resources

4.5.2 z/VM - necessary definitions

The resources listed in Table 2 on page 217 lists the resources you have to define to the virtual machine in which you intend to bring up your z/OS image. Please observe that the table does not contain all the necessary details, for instance all the DASD disk device addresses.

4.5.3 Define z/OS guest in USER DIRECT

After having obtained the necessary data about the z/OS image configuration, we logged on our z/VM MAINT userid and updated the z/VM USER DIRECT file (see Chapter 2.2.4, “User Directory” on page 22 and Chapter , “USER DIRECT file” on page 294).

> **Note:** It might make sense to name your guest machines using the corresponding z/OS sysname

Figure 4-27 shows definitions for the z/OS guest.

```
USER SC59 GUESTZOS 2G 4G G                              1
 INCLUDE IBMDFLT                                        2
 ACCOUNT ITS30000                                       3
 IPL CMS PARM AUTOCR                                    4
 MACHINE ESA 4                                          5
 OPTION MAINTCCW QUICKDSP                               6
 CONSOLE F008 3215                                      7
NICDEF 2000 TYPE QDIO DEVICES 6 LAN SYSTEM VSW1         8
MDISK 0191 3390 2390 2 LX6W02 MR                        9
MDISK 813A 3390 0 END OP1TSA MWV                       10
MDISK 8244 3390 0 END OP1HFC MWV
....
MDISK C730 3390 0 END IODFPK RR                        11
MDISK D42A 3390 0 END Z19RF1 RR
```

Figure 4-27 USER DIRECT entry for the SC59 user id.

Description of the statements from the line numbering in Figure 4-27:

1 Establishes USER *SC59* with a password of *GUESTZOS*. It is defined with 2 GB storage initially and 4GB as a maximum, and defined as a privilege class G user (no special privileges)

2 Includes the PROFILE IBMDFLT (not shown here)

3	Accounting information.
4	IPLs CMS on logon, which automatically executes the PROFILE EXEC, if present. From the PROFILE EXEC, you could execute other execs.
5	Defines the mode for the guest. For this particular user, the architecture is ESA with up to 4 CPUs
6	Establishes options for the guest user ID. MAINTCCW implies that the guest is allowed to initialize any DASD it uses. QUICKDSP causes the guest to be added to the dispatch list immediately when it has work to do without entering the eligible list.
7	Guest console address defined as the same address as the z/OS console. Note that it has to be defined as 3215 at this stage due to z/VM console requirements. It will be redefined to 3270 in order to IPL z/OS in the IPLSC59 EXEC (see Figure 4-30 on page 223)
8	Defined a connection to a z/VM vswitch as an OSA device with six consecutive addresses starting at address 2000.
9	Defines the guest's minidisk in CMS. Used for CMS file system, for instance PROFILE EXEC (described later)
10 - 11	Definitions of the DASD that z/OS requires. They are all defined as minidisks, starting in cylinder 0 and ending in the highest cylinder (the entire disk). In our case, the real address and the virtual address of every device is identical. The disk addresses could be defined using addresses other than the real ones to simplify configuration administration of test images. Note also that the disk is defined having an access mode of MWV, meaning Multi-Write access. The last V defines that CP should use its virtual reserve/release support for I/O operations for this minidisk. This is to ensure integrity if accessed from several guests. If there are shared read only DASD, then access can be defined as RR.

Note: Not all disks are listed, just a few examples to illustrate. There are additional minidisks defined to guest *SC59*.

> **Tip:** It is possible to define minidisks with access mode of RR to prevent them from being updated, if that should be desirable. This could be useful for preventing allocations or updates to certain disks (for instance your SYSRES or IODF disk) without adjusting any access rules in RACF or a similar product.

Once the updates to the USER DIRECT file are done, save the file and issue the DIRECTXA command to load the new directory into z/VM storage.

```
directxa user direct c
```

4.5.4 ZOS1 guest - console considerations

In this scenerio, we choose not to use any channel attached console to operate the image for simplicity. This implied that we could log on to the z/VM guest userid, IPL the z/OS, and use that session as a z/OS operator console.

Note that there is a concerns regarding this mode of operation:

- ***If the z/OS operator issues a CP LOGoff, the z/OS system will immediately terminate.*** A CP DISConnect should be used to keep the system running.

> **Important:** When running z/OS as a z/VM guest, the HMC console can not be used for IPLing the z/OS guest

This type of operation also implies that we need to change the mode for the operator console to 3270 prior to IPLing z/OS to avoid console lockouts, as 3270 is the supported terminal type in z/OS.

4.5.5 The SC59 guest - creating a PROFILE EXEC

We created a PROFILE EXEC file for guest user ID SC59 to define additional CPUs. Other definitions could also be added according to each installation’s needs. The PROFILE EXEC is executed each time an IPL of CMS is performed. This exec is also executed when a userid is autologged onto the z/VM system. Reconnecting from a DISCONNECTed a virtual machine will not execute the PROFILE EXEC.

Figure 4-28 on page 221 shows the PROFILE EXEC file for the SC59 user id.

```
 PROFILE  EXEC     F1  V 130  Trunc=130

00000 * * * Top of File * * *
00001 /* PROFILE EXEC FOR SC59 */
00002 set pf12 ret
00003 'CP DEFINE CPU 1'
00004 'CP DEFINE CPU 2'
00005 'CP DEFINE CPU 3'
```

Figure 4-28 PROFILE EXEC for SC59

Important: By defining the PROFILE EXEC like we did, you could not AUTOLOG this machine and have the z/OS system IPL automatically. The file does not contain any statements to actually IPL the z/OS image.

Instead of placing the statements and commands necessary to perform an IPL of the z/OS image in the profile, we created an exec to actually perform this. In short, this exec does three things:

1. Switches the mode of the console to 3270
2. Issues command SET RUN ON
3. IPLs the z/OS image

The contents of the *IPLSC59 EXEC* is shown in Figure 4-29 on page 222

```
 IPLSC59 EXEC     A1  V 130  Trunc=130 Size=14 Line=0 Col=1 Alt=2

* * * Top of File * * *
/* Exec to IPL the SC59 guest system */
trace 'o'
'vmfclear'
cmd = 'TERM CONMODE 3270'||'15'X
cmd = cmd||'SET RUN ON'||'15'X
say '**********************************************'
say '*                                            *'
say '*   This is a z/OS 1.9 system,               *'
say '*   WAIT, the system is coming up... SC59    *'
say '*                                            *'
say '**********************************************'
  cmd = cmd||'IPL D42A LOADPARM C73059M1 CLEAR'
push cmd
exit
* * * End of File * * *
```

Figure 4-29

Explanation of the significant statements in the IPLSC59 exec:

TERM CONMODE 3270 Switches the mode (or terminal type) to 3270

SET RUN ON This statement is used to avoid the virtual machine entering CP read mode causing the z/OS to halt

IPL D42A.... This is the actual IPL command for z/OS as normally entered via the HMC if you are running z/OS in a LPAR

The ‘15’x coded after each command, is the end of line indicator. The commands themselves are stacked and they are issued after the exec has ended (not controlled by any logic in the exec itself).

> **Note:** The important statements or commands in the *IPLSC59 EXEC*, the **TERM CONMODE 3270**, the **SET RUN ON** and the IPL D42A LOADPARM C73059M1 CLEAR statements, *could* all be issued as separate commands from the virtual machine itself, rather than executing the *IPLSC59 EXEC*.

4.5.6 IPLing the system as a z/VM guest

After having created the above mentioned files, we took the z/OS system running in an LPAR, down. Then we logged on to z/VM userid, **SC59,** and ran the **IPLSC59 EXEC**. After a short while (like a normal IPL of a LPAR would), we saw the initial messages and our session appeared identical to a normal z/OS operator console. After we brought all the necessary started tasks and sub systems up, we issued a **CP DISC** to disconnect from the system console, leaving the z/OS system running.

> **Note:** One 'peculiarity' of our way of defining the console, was that on the initial IPL, we had to prefix operator responses with**#CP VINPUT VMSG** in order to communicate with z/OS when in **NIPCON** mode (for instance **#CP VINPUT VMSG R 00,CONTINUE**). After entering the normal console mode, this was no longer true. This should not be necessary when operating physically attached consoles.

4.5.7 z/OS concerns

One thing to be aware of from a z/OS perspective, is any use of symbolics **HWNAME** and LPARNAME. If these are used in PARMLIB member (for instance your **IEASYMxx**) or in member **LOADxx** residing in PARMLIB or SYSn.IPLPARM, they would need your attention. The LPARNAME will never be resolved when running z/OS as a z/VM guest, so any filtering on values of these variables will fail. Depending on your incorporated logic, the IPL could fail due to this issue. If you have filters concerning **HWNAME**, you should examine if you need to change it to **VMUSERID** instead. For details, refer z/OS: MVS Initialization and Tuning Reference, SA22-7592.

4.5.8 Adding a 3390 disk to the z/OS guest

As with any z/OS system running in an LPAR, there are occasions when additional I/O devices, such as DASD, will be needed by the system. In an LPAR, defining the new DASD in the IOCDS for that LPAR will make the new volumes available to the z/OS system. There is a similar capability when running a z/OS system as a z/VM guest. After the z/OS guest system was brought up, we went through a series of steps to bring a new 3390 disk volume online to the guest system without interrupting the system or requiring an IPL. We also decided to bring the *physical* device address 8222 online to the z/OS guest as device address 6000 as defined on z/OS. The disk had a volser of DK8222. The first thing we did was to add it as a minidisk for the guest user (in USER DIRECT file), as illustrated in Figure 4-30 on page 223

```
....
MDISK 6000 3390 0 END DK8222 MWV
```

Figure 4-30 USER DIRECT statement for z/OS guest user minidisk

Issuing the DIRECTXA command loads the directory to z/VM storage

After having added the volume, while still logged on to MAINT, issued the command to make the new DASD volume available to the z/VM system and use by guest users:

ATT 8222 SYSTEM

After that, we logged on to the guest z/OS userid on z/VM. As this also acted as a operator console (see our assumption in Chapter 4.5.5, “The SC59 guest - creating a PROFILE EXEC” on page 220 , we pressed **PA1** (or the keystrokes simulating it from our 3270 emulator) to enter CP read mode. From there, we issued the command to make the new volume available to the z/OS system:

LINK * 6000 6000 MWV as illustrated in Figure 4-31

```
00: link * 6000 6000 mwv
00: DASD 6000 LINKED R/W
```

Figure 4-31 LINK command issued from z/OS guest user id

After clearing the CP screen thus entering the z/OS operator console again, we observed this message.

IOS1501 DEVICE 6000 NOW™ AVAILABLE FOR USE

From the z/OS system console, we issued a **D U,,,6000,1** to verify that the status had not changed (caused by the fact that we now were running under z/VM). Then we varied the device online from z/OS, see snapshot of the z/OS system log in Figure 4-32 on page 224.

```
- 12.57.20   IOS1501 DEVICE 6000 NOW AVAILABLE FOR USE
- 12.57.58  d u,,,6000,1
      12.57.58  IEE457I 12.57.58 UNIT STATUS 147
      UNIT TYPE STATUS        VOLSER     VOLSTATE
      6000 3390 F-NRD                        /RSDNT
    - 12.58.10  v 6000,online
      12.58.10  IEE302I 6000     ONLINE
  00- 12.58.24  d u,,,6000,1
      12.58.24  IEE457I 12.58.24 UNIT STATUS 153
      UNIT TYPE STATUS        VOLSER     VOLSTATE
      6000 3390 O             DK8222     PRIV/RSDNT
```

Figure 4-32 z/OS SYSLOG

This demonstration could give you some ideas on how you could use z/VMs virtualizition capabilities to manipulate device addresses as seen by z/OS. A possible scenario could be to have a relatively static hardware configuration (considering the IOCDS) for test z/OS images. By cloning a z/OS preconfigured system, you would have the possibility to manipulate physical device addresses and their corresponding virtual addresses for each guest in z/VM, thus reducing the need for configuration changes.

5

z/VM System operations

In this chapter we introduce some of the most common systems operations tasks for z/VM and Linux. In addition, we compare typical z/OS system operations with z/VM and Linux.

Objectives

On completion of this chapter, you should be able to:

- Communicate with other z/VM users
- Understand the IPL and shutdown processes of z/VM
- Perform basic management of z/VM virtual machines
- Perform resource management
- Run processes similar to batch jobs
- Get pointers to cloning systems

5.1 Using a Console communications to send messages to other users

Chapter 2.7, “Consoles” on page 55 discusses the types of consoles that z/VM has available, using these consoles allows communication to other z/VM users.

You can communicate with z/VM users by sending a:

- General information message to all logged-on users
- General information message to a specific user
- Warning message to all logged-on users
- Warning message to a specific user

z/OS analogy: z/OS users make use of the `SEND` command

Notes:

- The **`MESSAGE`** and **`WARNING`** commands are limited by the length of the input command area. If the entire text of a message does not fit in this area, enter another command with the remaining text.
- If an external security manager (ESM) is installed, you may not be authorized to use the **`MESSAGE`** or **`WARNING`** commands. However, messages sent to or from the system operator and messages sent with the `ALL` option are not subject to authorization checking by the ESM. For additional information, contact your security administrator.
- If CP does not issue your message the way you entered it, it may be because you are including special line-editing symbols in the text of your message. For example, if logical line editing is in effect, the # symbol is your logical line-end symbol; if you include a # in your message text, CP cuts off your message. To prevent CP from interpreting these symbols as logical line-editing functions, enter:
  `SET LINEDIT OFF`

  CP then issues your messages as you enter them.

5.1.1 Sending a General Information Message to All Users

To send a general information message, such as “Query log for weekend schedule,” to all logged-on users, enter:

```
message all query log for weekend schedule
```

or, when Cross System Extensions[1] (CSE) is active and you want all users logged on all systems in the CSE complex to receive the message, enter:

```
message all at all query log for weekend schedule
```

The following message appears on the display screen of all users able to receive the message:

```
hh:mm * MSG FROM OPERATOR: QUERY LOG FOR WEEKEND SCHEDULE
```

where *hh:mm* indicates the time the message was sent. Each user receives the message right away unless:

- An action is pending at the user's display In this case, the user receives the message when the action completes.
- The user is running in full-screen mode or has entered the **SET MSG OFF** command. In these cases, the user does not receive the message and you receive an error message.
- The user is not logged on.

5.1.2 Sending a general information message to a Specific User

Sometimes you need to send only a general information message to a specific user. For example, if you need to inform user VMUSER1 that he is not authorized to use pack RC015, which he just asked you to mount, enter the following command:

```
message vmuser1 you are not authorized to use pack rc015
```

As soon as user VMUSER1 has no action pending at his display, the message appears on his display screen (provided he is not running in full-screen mode or has not entered the **SET MSG OFF** command). Only `VMUSER1` receives this message.

5.1.3 Sending a Warning Message to All Users

Occasionally, you may have a system problem that requires you to take some action, such as needing to IPL the system, that would disrupt a user's virtual machine operation. When this happens, using the warning command will immediately interrupt the user's console with the urgent message, unlike the `MESSAGE ALL`, to inform all users of the situation:

[1] Reference on CSE in *z/VM CP Planning and Administration*: http://publibz.boulder.ibm.com/epubs/pdf/hcsg0b21.pdf

```
warning all the system will be IPLed in 5 monutes, please log off.
```

or, when CSE is active and you want all users logged on all systems in the CSE complex to receive the message, enter

```
warning all at all the system will be IPLed in 5 monutes, please log
off.
```

In response, CP displays the message on all users' display screens within the next 60 seconds (provided they are not running in full-screen mode and have not entered the **SET WNG OFF** command). This should give them time to save what they are currently working on.

> **Note:** If you use the **MESSAGE** command to send a warning, a user with an action pending does not receive it until that action completes, which may be too late.

5.1.4 Sending a Warning Message to a Specific User

Sometimes you may have to send a high-priority message to a specific user. For example, user MVSOPR1 may submit a high-priority file that needs to be processed on printer 00E, but the printer is dedicated to user VMUSER1. Before you reassign the printer to user MVSOPR1, you may want to send user VMUSER1 the following warning message:

```
warning vmuser1 system needs printer 00e - draining in 5 minutes
```

If he is not running in full-screen mode and has not entered the **SET WNG OFF** command, user VMUSER1 receives this message within the next 60 seconds and should stop sending files to this printer for processing.

5.2 Running the system

This section explains what happens in your z/VM system when it is brought up and when it is taken down. It is good to have some knowledge of the IPL and shutdown process for systems monitoring and problem debugging.

5.2.1 z/VM IPL workflow

When you IPL your z/VM system, many steps need to take place during the IPL before it is fully functional and ready for use. Figure 5-1 on page 231 shows the flow of these steps.

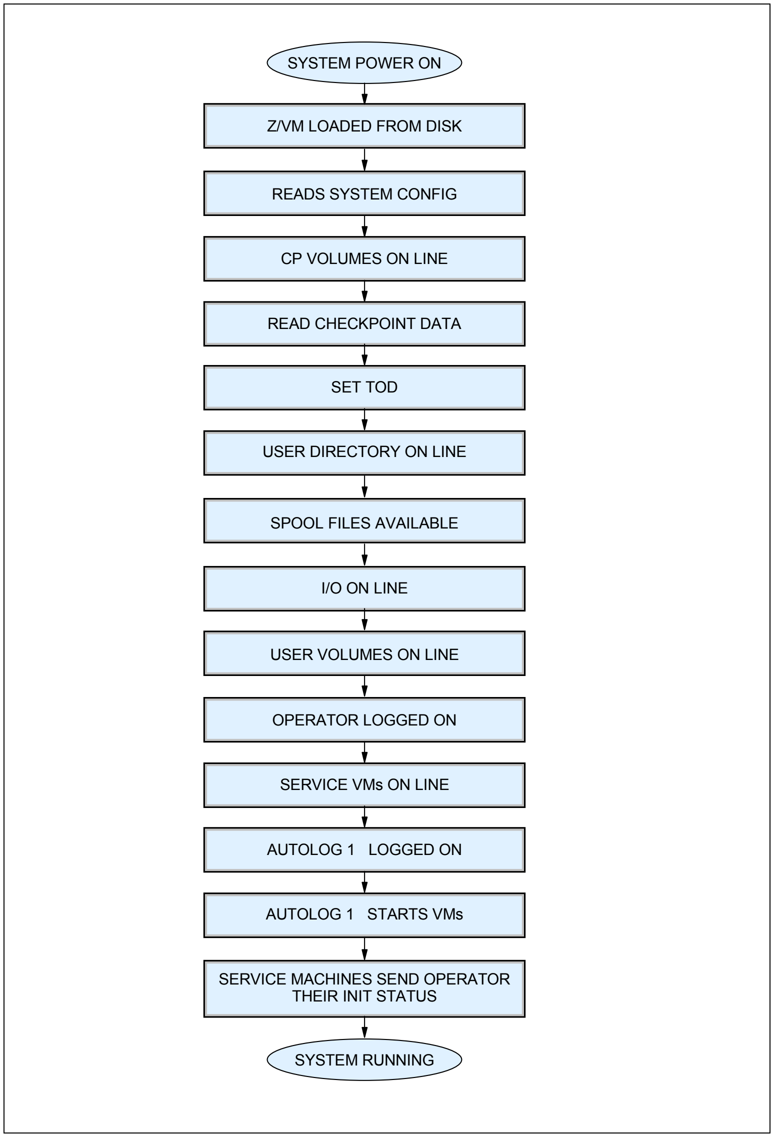

Figure 5-1 z/VM IPL flowchart.

1 The z/VM operating system is loaded from disk. This comprises the system CP nucleus.

2 The CP reads the `SYSTEM CONFIG` file to identify configuration parameters and implement them.

3 The operating system brings the CP owned volumes online, such as the ones used for paging, spool, and t-disk.

4 Depending on how it was started (see “Warm Starts, Force Starts, Cold Starts, and Clean Starts” on page 234), z/VM reads the information about systems attributes saved in the checkpoint and warmstart data files[2] during last shutdown and restores them.

5 System prompts for set up of time-of-day (TOD).

6 The user directory, with the users definitions, is brought online.

7 Existing spool files are made available.

8 All connected I/O is brought online and made available.

9 User volumes defined in `SYSTEM CONFIG` are made available.

10 The `OPERATOR` user is logged on.

11 The `OPERATOR` virtual machine starts the logging/accounting/symptom services by logging on the `erep`, `diskacnt` and `opersymp` users.

12 The `OPERATOR` userid autologs `AUTOLOG1`.

13 `AUTOLOG1` starts any other virtual machine that it has been instructed to start accordingly to its `PROFILE EXEC`.

14 Services machines send messages to `OPERATOR` on initialization status.

Example 5-1 below shows the output of a z/VM IPL process. Interactions with the console are in **bold**.

Example 5-1 z/VM IPL process.

```
14:58:30 z/VM  V5 R3.0  SERVICE LEVEL 0703 (64-BIT)
14:58:31 SYSTEM NUCLEUS CREATED ON 2008-05-23 AT 10:40:19, LOADED
FROM LX6RES
14:58:31
14:58:31 ****************************************************************
14:58:31 * LICENSED MATERIALS - PROPERTY OF IBM*                       *
14:58:31 *                                                             *
14:58:31 * 5741-A05 (C) COPYRIGHT IBM CORP. 1983, 2007. ALL RIGHTS     *
14:58:31 * RESERVED. US GOVERNMENT USERS RESTRICTED RIGHTS - USE,      *
14:58:31 * DUPLICATION OR DISCLOSURE RESTRICTED BY GSA ADP SCHEDULE    *
```

[2] This file is stored in the system RES volume

```
14:58:31 * CONTRACT WITH IBM CORP.                                    *
14:58:31 *                                                            *
14:58:31 * * TRADEMARK OF INTERNATIONAL BUSINESS MACHINES.             *
14:58:31 *****************************************************************
14:58:31
 14:58:31 HCPZCO6718I Using parm disk 1 on volume LX6RES (device 1B34).
 14:58:31 HCPZCO6718I Parm disk resides on cylinders 39 through 158.
14:58:31 Start ((Warm|Force|COLD|CLEAN) (DRain) (DIsable)  (NODIRect)
14:58:31       (NOAUTOlog)) or (SHUTDOWN)
14:58:59 WARM
14:58:59 NOW 14:58:59 EDT THURSDAY 2008-06-05
14:58:59 Change TOD clock (Yes|No)
14:59:03 NO
14:59:03 The directory on volume LX6RES at address 1B34 has been
brought online.
14:59:05 HCPWRS2513I
14:59:05 HCPWRS2513I Spool files available      376
14:59:07 HCPWRS2512I Spooling initialization is complete.
14:59:07 DASD 1B35 dump unit CP IPL pages 17646
14:59:07 HCPAAU2700I System gateway VMGUEST identified.
14:59:07 z/VM Version 5 Release 3.0, Service Level 0703 (64-bit),
14:59:07 built on IBM Virtualization Technology
14:59:07 There is no logmsg data
14:59:07 FILES: 0031 RDR, 0009 PRT,   NO PUN
14:59:07 LOGON AT 14:59:07 EDT THURSDAY 06/05/08
14:59:07 GRAF  0020 LOGON  AS  OPERATOR USERS = 1
14:59:07 HCPIOP952I 2048M system storage
14:59:07 FILES: 0000314 RDR, 0000030 PRT, 0000001 PUN
14:59:07 HCPCRC8082I Accounting records are accumulating for userid
DISKACNT.
14:59:07 XAUTOLOG EREP
14:59:07 Command accepted
14:59:07 XAUTOLOG DISKACNT
14:59:07 Command accepted
14:59:07 XAUTOLOG AUTOLOG1
14:59:07 Command accepted
14:59:07 XAUTOLOG OPERSYMP
14:59:07 Command accepted
14:59:07 AUTO LOGON  ***      EREP     USERS = 2     BY OPERATOR
14:59:07 AUTO LOGON  ***      DISKACNT USERS = 3     BY OPERATOR
14:59:07 AUTO LOGON  ***      AUTOLOG1 USERS = 4     BY OPERATOR
14:59:07 AUTO LOGON  ***      OPERSYMP USERS = 5     BY OPERATOR
14:59:07 HCPCLS6056I XAUTOLOG information for EREP: The IPL command is
verified by the IPL command processor.
14:59:07 HCPCLS6056I XAUTOLOG information for DISKACNT: The IPL command is
verif
ied by the IPL command processor.
14:59:07 HCPCLS6056I XAUTOLOG information for AUTOLOG1: The IPL command is
verif
```

```
ied by the IPL command processor.
14:59:07 HCPCLS6056I XAUTOLOG information for OPERSYMP: The IPL command is
verif
ied by the IPL command processor.
14:59:07 AUTO LOGON  ***      VMSERVS  USERS = 6     BY AUTOLOG1
14:59:07 AUTO LOGON  ***      VMSERVU  USERS = 7     BY AUTOLOG1
14:59:07 AUTO LOGON  ***      VMSERVR  USERS = 8     BY AUTOLOG1
14:59:07 USER DSC   LOGOFF AS  AUTOLOG1 USERS = 7
14:59:07 AUTO LOGON  ***      DTCVSW1  USERS = 8     BY AUTOLOG1
14:59:07 AUTO LOGON  ***      DTCVSW2  USERS = 9     BY AUTOLOG1
14:59:07 AUTO LOGON  ***      TCPIP    USERS = 10    BY AUTOLOG1
14:59:07 * MSG FROM EREP :1 RECORDING FILE(S), 40 RECORDS, A DISK 03 % FU
LL
14:59:07 * MSG FROM DISKACNT:17 RECORDING FILE(S),1097 RECORDS, A DISK 27 %
FULL
14:59:07 HCPCRC8064I Recording data retrieval has been started; recording
*LOGREC for userid EREP.
14:59:07 HCPCRC8064I Recording data retrieval has been started; recording
*ACCOUNT for userid DISKACNT.
 14:59:07  * MSG FROM OPERSYMP: 26 RECORDING FILE(S), 26 RECORDS, A DISK 04
% FULL
14:59:07 HCPCRC8064I Recording data retrieval has been started; recording
*SYMPTOM for userid OPERSYMP.
14:59:08 3020-3022 ATTACHED TO TCPIP BY TCPIP
14:59:08 AUTO LOGON  ***      FTPSERVE USERS = 11    BY TCPIP
```

5.2.2 Warm Starts, Force Starts, Cold Starts, and Clean Starts

z/VM provides you with four basic types of starts: *warm*, *force*, *cold*, and *clean*. The difference among them is in how much of the system's environment CP restores, with a warm start restoring the most and a clean maintaining the least.

If a warm start fails, contact your system support personnel before you use a force or cold start since critical files could be lost.

In order to restore the system's environment, you must save it first. This is done whenever you issue a **SHUTDOWN command** and the response, **SHUTDOWN COMPLETE** is received. When you do so, CP saves the following parts of the system's environment:

- Any accounting, EREP, and symptom records in storage.

  Accounting, EREP, and symptom record recording service virtual machines normally transfer these records from storage to disk. If these virtual machines are not retrieving, the records remain in storage. The **SHUTDOWN command** will write any records that are still in storage into the checkpoint area on DASD so that during the next IPL the records can be returned to storage for retrieval.

- System log message.

  The system log message communicates information to users when they log on or reconnect.

- Spool file queues.

  Spool files are collections of data on disk waiting to be processed by real or virtual readers, printers, or punches. The **SHUTDOWN** command saves information about the queues that locate these spool files.

  Note: Spool files may not be restored in the same order following a warm or force start.

  z/OS analogy: the z/VM **SHUTDOWN** command is similar to the **HALT** (Z EOD) command, with the exception that z/OS does not keep warm data.

- The system data file queues.

  The system data file queues hold the following collections of data, called **system data files**: named saved systems (NSSs), discontiguous saved segments (DCSSs), image libraries, user class restructure (UCR) files, message repository files, and system trace files.

  – A **named saved system (NSS)** is a copy of an operating system that a user has named and retained in a system data file. The user can load the operating system by its name, which is more efficient than loading it by device number.

  – A **discontiguous saved segment (DCSS)** refers to one or more pages of storage that a user has named and retained in a system data file. When a discontiguous saved segment, defined as shareable, is loaded, more than one user can access it.

  – An **image library** is a set of modules, contained in a system data file, that define the spacing, characters, and copy-modification data that a printer uses to print a spool file.

  – **User class restructure (UCR)** files expand the privilege classes of commands and DIAGNOSE codes. This lets your installation customize the privilege class structure of z/VM.

  – **Message repository** files contain z/VM messages and responses translated into a national language.

  – **System trace** files contain records of events that occur within the system. Use these files to determine the source of problems in the system.

The **SHUTDOWN** command saves information about the queues that locate these system data files.

- The status of unit record devices, displays, and 3270 printers.

When you enter the **SHUTDOWN** command, you save all these parts of the system's environment. Then, when you bring the system back up with a warm start, CP restores them.

If you enter **SHUTDOWN REIPL**, CP attempts to do an automatic start of the specified module.

See the *z/VM: CP Commands and Utilities Reference*[3] book for other restrictions when you re-IPL another module.

Unless you have a reason to do otherwise, specify a warm start. Keep in mind that if you do not shut down the z/VM system with the SHUTDOWN command, you may not be able to bring it back up with a warm start. The information that the **SHUTDOWN** command saves is what allows a warm start to work.

During a force start, CP tries to restore most of what it restores during a warm start, but it may not be able to do so completely. Just as in a warm start, CP tries to restore, in this order:

- Accounting, EREP, and symptom records in storage
- System log message
- Spool file and system data file queues.

However, if CP encounters an error during the first step, it immediately goes on to recover the spool file and system data file queues. Unless you specified `FEATURES ENABLE LOGMSG_FROM_FILE` in the system configuration file, the system log message is lost. Any remaining accounting, EREP, and symptom records are lost. Further, if CP encounters an error while recovering a spool file, it immediately goes on to the next file in the queue, and the file on which the error occurred is scheduled to be deleted. When the preliminary phase of spooling initialization is complete, a message providing spool file summary status is displayed, and the operator can then stop system initialization without the loss of any spool files.

Also, a force start does not restore any of the unit record device characteristics, such as the class to be processed or the image library. Instead, a force start tries to start the unit record devices with the default characteristics specified on the `RDEVICE` statement in the system configuration file. A force start does try to restore the status of all displays and 3270 printers.

[3] http://publib.boulder.ibm.com/infocenter/zvm/v5r3/index.jsp?topic=/com.ibm.zvm.v53.hcpb7/hcse4b21.htm

Therefore, after a force start, CP may not have completely recovered all of the system's environment. Use a force start only when authorized by the system support personnel after a warm start fails.

If a force start fails, you must perform a *cold* start. During a cold start, CP tries to recover *only* the system data files. This means that when you perform a cold start, you lose all of your spool files, all accounting, EREP, and symptom records in storage. The system log message is lost if you did not specify the `FEATURES ENABLE LOGMSG_FROM_FILE` statement in the system configuration file. The status of your unit record devices, displays, and 3270 printers may also differ from their status before the restart. Use a cold start only when both a warm start and a force start fail.

The essential points to grasp about warm starts, force starts, cold starts, and clean starts are:

- To save the system's environment before you restart it, use the **SHUTDOWN** command.
- To restore the system's environment, bring the system back up with a warm start.
- If a warm start fails,
  - Check the DASD volumes.
  - Contact system support personnel.
  - When repeated attempts to perform a warm start fail, try a force start to restore as much of the system's environment as possible.
- If you decide to continue with system initialization even though there are files scheduled to be deleted, remember that those files cannot be recovered.
- If a force start fails, try a cold start.
- A clean start IPLs the system without attempting to recover spool files and system data files that existed prior to system shutdown.

5.2.3 Checking system resources

Once your system has started, you will want to check your system resources. This section discusses various commands that will assist you in checking your system resources.

The following section show a list of commands that will assist you in checking the availability of the system.

Checking system level: `query cplevel` and `query cmslevel`

These commands can be used to check:

- The software version level, release level, and release modification level
- The software service level number
- The date and time (translated to the current active time zone) that CP system software was written to DASD.
- The date and time CP was last started
- The release and service level of CMS.

This can be useful to ensure a maintenance has been correctly applied on a system.

Example 5-2 Checking CP and CMS levels

```
q cplevel
z/VM Version 5 Release 3.0, service level 0701 (64-bit)
Generated at 05/29/07 23:21:12 EDT
IPL at 06/11/08 07:58:34 EDT
Ready; T=0.01/0.01 11:45:20
q cmslevel
CMS Level 23, Service Level 701
Ready; T=0.01/0.01 11:45:24
```

Checking CP load parameters: `query cpload`

Use **QUERY CPLOAD** to display information regarding the last CP IPL. The information displayed includes the location of the CP module that was last used, the location of the parm disk, and how CP was started.

Example 5-3 Querying CPLOAD

```
q cpload
Module CPLOAD was loaded from minidisk on volume LX6RES at cylinder 39.
Parm disk number 1 is on volume LX6RES, cylinders 39 through 158.
Last start was a system IPL.
Ready; T=0.01/0.01 11:46:07
```

Checking the amount of real storage: `query storage`

The **QUERY STORAGE** command (abbreviated as **Q STO**) is used to display the amount of real memory that is available to a LPAR once started.

Example 5-4 Checking the amount of real storage available in the LPAR

```
q stor
```

```
STORAGE = 4G
Ready; T=0.01/0.01 11:46:44
```

Checking the amount of expanded storage: `query xstorage`

This command, abbreviated as **Q XSTO**, is used to display the amount of real expanded storage configured in a partition. The output of the command also show the amount of expanded storage in used, as well as the amount used for minidisk caching (MDC).

Example 5-5 Checking the amount of Expanded Storage available in the LPAR

```
q xstor
XSTORE= 2048M online= 2048M
XSTORE= 2048M userid= SYSTEM usage= 0% retained= 0M pending= 0M
XSTORE MDC min=0M, max=0M, usage=0%
XSTORE= 2048M userid=  (none)  max. attach= 2048M
Ready; T=0.01/0.01 11:47:27
```

Checking all processors are available: `query processors`

The **QUERY PROCESSORS** command, abbreviated as **Q PROC**, is used to display the current assignment of processors of the partition.

MASTER processor is the base processor, used for IPL. ALTERNATE processors are automatically brought online at IPL, while STANDBY are processors that are available to the partition, but offline.

The last field indicates weather a processor is a CP, an IFL, ICF, ZIIP or ZAAP.

Example 5-6 Checking the numbers and types of CPUs available to the LPAR

```
q processors
PROCESSOR 00 MASTER CP
PROCESSOR 01 ALTERNATE CP
PROCESSOR 02 ALTERNATE CP
PROCESSOR 03 ALTERNATE CP
PROCESSOR 04 STANDBY CP
PROCESSOR 05 STANDBY CP
Ready; T=0.01/0.01 11:51:31
```

Changing the availability of processors to the system

Under normal circumstances, you bring up and run the system with all the processors in the processor complex or partition. But if support personnel at your installation instruct you to do so, you may use the **VARY ONLINE PROCESSOR** or **VARY OFFLINE PROCESSOR** commands to control whether a processor is available to z/VM. For more information on these commands, see the *z/VM: CP commands and Utilities Reference* book[4].

Checking the system DASD configuration: `query alloc`

After the system has IPLed, a good practice is to check the required paging and spooling are available and online.

The **`QUERY ALLOC PAGE`** is used to retrieve paging space configuration. In Example 5-7 on page 240, four 3390-03 (each are 3339 cylinders) have been configured as paging devices. Only 13 pages in total are allocated on these paging devices.

Example 5-7 Checking the paging space available in the LPAR

```
q alloc page
                EXTENT     EXTENT  TOTAL  PAGES   HIGH    %
VOLID  RDEV      START        END  PAGES IN USE   PAGE USED
------ ---- ---------- ---------- ------ ------ ------ ----
LX6PAG 1A22          1       3338 600840      0      0   0%
DK8226 8226          0       3338 601020      0      0   0%
DK8227 8227          0       3338 601020     13     26   1%
DK8228 8228          0       3338 601020      0      0   0%
                                  ------ ------        ----
SUMMARY                            2348K     13          1%
USABLE                             2348K     13          1%
Ready; T=0.01/0.01 11:54:40
```

Similarly, the **`QUERY ALLOC SPOOL command`** is used to check the spooling configuration. In Example 5-8 on page 240, only one 3390-03 DASD is defined as spooling device, and 140075 pages are in use, representing around 23% of the total pages available for spooling.

Example 5-8 Checking the spool space

```
q alloc spool
                EXTENT     EXTENT  TOTAL  PAGES   HIGH    %
VOLID  RDEV      START        END  PAGES IN USE   PAGE USED
------ ---- ---------- ---------- ------ ------ ------ ----
LX6SPL 1A21          1       3338 600840 140075 145430  23%
                                  ------ ------        ----
SUMMARY                           600840 140075         23%
USABLE                            600840 140075         23%
Ready; T=0.01/0.01 11:55:39
```

[4] http://publibz.boulder.ibm.com/epubs/pdf/hcse4b21.pdf

Finally, the **QUERY ALLOC TDISK**, displays all defined and assigned temporary disk space.

Example 5-9 Checking the amount of TDISK allocated

```
q alloc tdisk
                EXTENT     EXTENT                         %
VOLID  RDEV      START        END  TOTAL IN USE   HIGH USED
------ ---- ---------- ---------- ------ ------ ------ ----
LX6TDK 8225          1       3338   3338     20     20   1%
                                  ------ ------        ----
SUMMARY                             3338     20          1% CKD
USABLE                              3338     20          1% CKD
Ready; T=0.01/0.01 11:06:01
```

Checking system dump space configuration: query dump

The **QUERY DUMP** command shows:

- The device type and unit or units assigned to receive CP abnormal termination dumps
- The current settings of the DUMP options.

Example 5-10 Querying the DUMP configuration

```
q dump
DASD 1A21 dump unit CP IPL pages 31437
Ready; T=0.01/0.01 11:57:09
```

Checking the recording status: query recording

The **QUERY RECORDING** command, abbreviated as **Q REC**, is used to check the status of:

- CP data collection for these records:
  - Accounting
  - EREP (errors recording)
- CP data collection for symptom records
- Active retrieval of these records.

In Example 5-11 on page 242, all recordings are active, using the defaults userids defined in z/VM: EREP userid for error recording, ACCOUNT for accounting information, and OPERSYMP for symptoms records gathering. The count field would show any unprocessed records.

Example 5-11 Status of recordings in z/VM

```
q rec
RECORDING    COUNT     LMT USERID   COMMUNICATION
EREP      ON  00000000  002 EREP     ACTIVE
ACCOUNT   ON  00000000  020 DISKACNT ACTIVE
SYMPTOM   ON  00000000  002 OPERSYMP ACTIVE
Ready; T=0.01/0.01 11:57:36
```

Checking real devices availability

Once z/VM has IPLed, the system operator check that all resources allocated and defined in the configuration have been brought online.

Disks devices: `query dasd`

The **`QUERY DASD ALL`** command, abbreviated as **`Q DA ALL`**, displays the list of all real disks active in the partition, their owner (CP, SYSTEM, or FREE if they are not attached), their label, and the number of links to MDISKs defined on the volume. The last line informs that there are no offline DASDs. **`QUERY DASD`** will display all the dasd in use by the system.

Example 5-12 Querying all disks attached to the system.

```
q da all
DASD 1A20 CP OWNED  LX6RES   80
DASD 1A21 CP OWNED  LX6SPL   1
DASD 1A22 CP OWNED  LX6PAG   0
DASD 1A23 CP OWNED  LX6W01   127
DASD 1A24 CP OWNED  LX6W02   4
DASD 1A25 CP SYSTEM NW1A25   1
DASD 1A26 CP SYSTEM NW1A26   1
DASD 1A27 CP SYSTEM NW1A27   1
DASD 1A28 CP SYSTEM NW1A28   1
DASD 1A29 CP SYSTEM NW1A29   1
DASD 1A2A CP SYSTEM NW1A2A   1
DASD 1A2B CP SYSTEM NW1A2B   1
DASD 1A2C CP SYSTEM NW1A2C   1
DASD 1A2D CP SYSTEM NW1A2D   1
DASD 8221 CP SYSTEM DK8221   1
DASD 8226 CP OWNED  DK8226   0
DASD 8227 CP OWNED  DK8227   0
DASD 8228 CP OWNED  DK8228   0
DASD D815 CP SYSTEM LXD815   2
DASD D816 CP SYSTEM LXD816   1
DASD 1B34 LX6RES   , DASD 1B35 LX6SPL   , DASD 1B36 LX6PAG   , DASD 1B37
LX6W01
```

```
DASD 1B38 LX6W02  , DASD 8222 DK8222  , DASD 8223 LX8223  , DASD 8224
DK8224
DASD 8225 LX8225  , DASD D817 LXD817  , DASD D818 LXD818  , DASD D819
LXD819
DASD D81A LXD81A  , DASD D81B LXD81B  , DASD D81C LXD81C  , DASD D81D
LXD81D
DASD D81E LXD81E  , DASD D81F LXD81F
An offline DASD was not found.
Ready; T=0.01/0.01 11:58:00
```

Magnetic tape drives: `query tape`

The **QUERY TAPES** command, abbreviated as **Q TA**, is used to display the status of the magnetic tape drives available to the system and assigned for use. **QUERY TAPES ALL** will display all tape drives available to the system. Checking the status of real tape drives.

Example 5-13 query tape

```
q tape
An active tape was not found.
Ready; T=0.01/0.01 11:58:29
```

Real Channel to channel adapters: query ctca

The **QUERY CTCA** command can be used to check the status of real channel-to-channel adapters available to the system and in use. **QUERY CTC ALL** will display all available CTC devices.

Example 5-14 Querying real CTC adapters

```
q ctc
An active CTCA was not found.
Ready; T=0.01/0.01 12:00:26
```

Real OSA devices: `query osa`

The **QUERY OSA ALL** command displays the list of real OSA devices. **QUERY OSA** will display the osa devices in use. For each devices, the following information is displayed:

- the virtual machine to which they are dedicated
- the virtual device number used in the virtual machine configuration to refer to the real OSA device
- the type of the device (HIPER, OSA)
- the CHannel Path IDentifier to use to access the device, as well as its type.

Example 5-15 Displaying the status of real OSA devices attached to the partition

```
q osa all
OSA  3020 ATTACHED TO TCPIP    3020 DEVTYPE OSA        CHPID 0C OSD
OSA  3021 ATTACHED TO TCPIP    3021 DEVTYPE OSA        CHPID 0C OSD
OSA  3022 ATTACHED TO TCPIP    3022 DEVTYPE OSA        CHPID 0C OSD
OSA  3024 ATTACHED TO DTCVSW2  3024 DEVTYPE OSA        CHPID 0C OSD
OSA  3025 ATTACHED TO DTCVSW2  3025 DEVTYPE OSA        CHPID 0C OSD
OSA  3026 ATTACHED TO DTCVSW2  3026 DEVTYPE OSA        CHPID 0C OSD
OSA  3028 ATTACHED TO DTCVSW1  3028 DEVTYPE OSA        CHPID 0C OSD
OSA  3029 ATTACHED TO DTCVSW1  3029 DEVTYPE OSA        CHPID 0C OSD
OSA  302A ATTACHED TO DTCVSW1  302A DEVTYPE OSA        CHPID 0C OSD
OSA  3023 FREE    , OSA  3027 FREE    , OSA  302B FREE    , OSA  302C
FREE
OSA  302D FREE    , OSA  302E FREE    , OSA  302F FREE
An offline OSA was not found.
OSA  302F is an OSA Agent
Ready; T=0.01/0.01 12:00:51
```

Displaying running users: `query names`

To check if the required users have been correctly auto-logged, a system operator can use the **QUERY NAMES** command, abbreviated as **Q N**. This command shows the list of virtual machines currently running in the system, and their status:

- DSC: the machine is running disconnected
- L0003: the machine is logged on

The VSM machine is the userid of the VTAM service machine, managing the TCPIP user.

Example 5-16 Displaying running users

```
q n
LNXKEN   - DSC , LNXGUI   - DSC , LNXCER   - DSC , FTPSERVE - DSC
TCPIP    - DSC , DTCVSW2  - DSC , DTCVSW1  - DSC , VMSERVR  - DSC
VMSERVU  - DSC , VMSERVS  - DSC , OPERSYMP - DSC , DISKACNT - DSC
EREP     - DSC , OPERATOR - SYSC, MAINT    -L0003
VSM     - TCPIP
Ready; T=0.01/0.01 12:02:00
```

network status - `ifconfig` and `netstat`

These networking commands reside on TCPMAINT's 592 disk and the system operator would need to link to that disk before issuing these commands.

The **ifconfig** command will show the status of your TCP/IP network, including I.P. address, netmask, and amounts of data transferred.

```
ifconfig
OSA3020  inet addr: 9.12.4.89 mask: 255.255.252.0
         UP BROADCAST MULTICAST MTU: 1500
         vdev: 3020 type: QDIO ETHERNET portname: UNASSIGNED
         ipv4 router type: NONROUTER ipv6: DISABLED
         cpu: 0 forwarding: ENABLED
         RX bytes: 35296690 TX bytes: 3354354
Ready; T=0.01/0.01 13:43:59
```

Figure 5-2 ifconfig command

The NETSTAT commands will show more detailed information on the system's network. Issuing **NETSTAT ?** will display all the command options along with a brief explanation. As an example, the **NETSTAT GATE,** will display the network gateway information.

```
netstat gate
VM TCP/IP Netstat Level 530

Known IPv4 gateways:

Subnet Address  Subnet Mask   FirstHop Flgs PktSz Metric Link
--------------  -----------   -------- ---- ----- ------ ------
Default         <none>        9.12.4.1 UGS  1500  <none> OSA3020
9.12.4.0        255.255.252.0 <direct> UT   1500  <none> OSA3020

Known IPv6 gateways: None
Ready; T=0.01/0.01 13:55:46
```

Figure 5-3 netstat gate command

Checking system workload: indicate

To have an overview of the use of the resources of the system, the **INDICATE LOAD** (abbreviated as **IND LOAD**) command can be used. When issued from a class B user, the output is as shown in Example 5-17 on page 246. The first line shows the average processor utilization, and the overall number of CPU in the system. Next three lines displays usage of expanded storage, minidisk cache and paging activity. Next four lines displays the number of users in each of the four queues.

The last lines displays the current processor utilization. The system shown in the example was obviously doing nothing.

Example 5-17 Overview of system resources utilization

```
ind load
AVGPROC-000% 04
XSTORE-000000/SEC MIGRATE-0000/SEC
MDC READS-000000/SEC WRITES-000000/SEC HIT RATIO-000%
PAGING-0/SEC STEAL-000%
Q0-00001(00000)                           DORMANT-00011
Q1-00000(00000)           E1-00000(00000)
Q2-00000(00000) EXPAN-001 E2-00000(00000)
Q3-00006(00000) EXPAN-001 E3-00000(00000)

PROC 0000-000% CP       PROC 0001-000% CP
PROC 0002-000% CP       PROC 0003-000% CP

LIMITED-00000
```

5.2.4 z/VM shutdown workflow

Whenever a z/VM system goes down there are some important tasks that take place. Some of them are the opposite as to the IPL tasks, but are complimentary. For example, saving the system attributes in the checkpoint data file. Figure 5-4 on page 246 depicts the whole process.

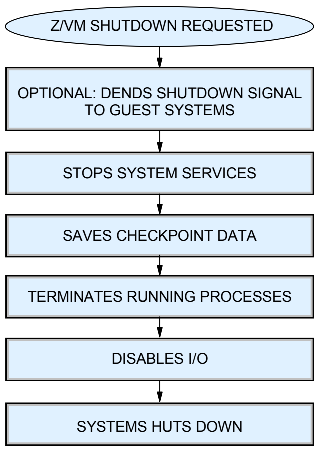

Figure 5-4 z/VM shutdown flowchart.

1	z/VM sends shutdown signals to guest systems (except for z/OS guests) requesting them to shut down. This allows guest system to cleanly shut themselves down before the entire system is shut off. This feature must be enabled in the SYSTEM CONFIG file (read "Starting virtual switches automatically at system startup" on page 179).
2	The system cleanly stops system services, such as logging and accounting for example.
3	z/VM saves the system attributes in the checkpoint data file on its `RES` volume.
4	All running processes are terminated.
5	z/VM disables of the I/O devices.
6	The running system is terminated.

Example 5-18 below shows the output of a z/VM shutdown process.

Example 5-18 z/VM shutdown process.

```
00: 14:54:42 HCPWRP963I SHUTDOWN STEP USOAC - JOURNAL USER TERMINATION
00: 14:54:43 HCPWRP963I SHUTDOWN STEP MFRSD - TERMINATE HARDWARE LOADER
00: 14:54:43 HCPWRP963I SHUTDOWN STEP APISD - TERMINATE OTHER PROCESSORS
00: 14:54:44 HCPWRP963I SHUTDOWN STEP ENASD - DISABLE TERMINAL DEVICES
00: 14:54:45 HCPWRP963I SHUTDOWN STEP ISHDN - SHUT DOWN I/O SUBSYSTEM
00: 14:54:45 HCPWRP963I SHUTDOWN STEP TTRAL - TERMINATE CONCURRENT COPY
SESSIONS
00: 14:54:46 HCPWRP963I SHUTDOWN STEP SVACV - ACTIVATE TERMINATION SAVE
AREAS
00: 14:54:46 HCPWRP963I SHUTDOWN STEP CHMOF - DISABLE CHANNEL MEASUREMENT
00: 14:54:47 HCPWRP963I SHUTDOWN STEP ISHDA - DISABLE ALL DEVICES
00: 14:54:47 HCPWRP963I SHUTDOWN STEP CKPSH - TAKE A CHECKPOINT
00: 14:54:48 HCPWRP963I SHUTDOWN STEP OPRCK - SAVE OPERATOR CONSOLE LIST
00: 14:54:48 HCPWRP963I SHUTDOWN STEP MCWMD - DETERMINE MACHINE CHECK
STATUS
00: 14:54:49 HCPWRP963I SHUTDOWN STEP SDVRS - RESET IBM DASD CU
CHARACTERISTICS
00: HCPWRP962I VM SHUTDOWN COMPLETED IN 10 SEC
00: 14:54:49 HCPWRP963I SHUTDOWN STEP SVADV - DEACTIVATE TERMINATION SAVE
AREAS
00: 14:54:50 HCPWRP961W SYSTEM SHUTDOWN COMPLETE
00: HCPGIR450W CP entered; disabled wait PSW 00020000 00000000 00000000
00000FFF
```

5.2.5 z/VM services

z/VM provides several services in the form of virtual machines. Unlike some operating systems, CP does not have the concept of a *process*. Instead, all CP knows is how to run virtual machines. This means that any extra service functionality not built directly into CP must be implemented as a service virtual machine.

A *service virtual machine* is the same as any other virtual machine, except it runs some software (typically on top of CMS) that provides a service to some or all of the other users on the system.

The following examples are services provided by service virtual machines:

- TCP/IP networking stack
- User directory maintenance (Dirmaint)
- Security manager (RACF)
- Performance data collection and reporting (Performance Toolkit)
- Systems management service (SMAPI)

Service virtual machines in z/VM usually have a user name that corresponds with the service it provides. Table 5-1 lists some of the common service virtual machines and brief descriptions of the service provided.

Table 5-1 Common service virtual machines

Guest name	Service provided
TCPIP	TCP/IP networking stack and tools like FTP and telnet.
DIRMAINT	User directory maintenance.
DATAMOVE	Provides disk copying and formatting services for DIRMAINT.
PERSFVM	Performance recording and reporting.
RACFVM	Security management for guests, devices and services.
VSMSERVE	Systems management service.
RSCS	Remote spool device capability.
PVM	Remote communication and system access.
EREP	Error recording.
DISKACNT	User accounting recording

Guest name	Service provided
OPERSYMP	Symptom recording
VMSERVR VMSERVS VMSERVU	Support for Shared File System (SFS).

5.3 Managing a guest operating system virtual machine

Guest support in z/VM allows you to run multiple images of production operating systems. You can use guest support in z/VM to develop, test, manage and migrate operating systems that run on System z alongside your production systems. In the following sections you learn about guest systems supported and how to operate the virtual machines.

5.3.1 Guest support

All of the supported mainframe operating systems can run under z/VM, including z/VM itself. For a list of supported mainframe operating systems and the type of support z/VM offers, along with other useful information, refer to notes in *z/VM General Information*, GC24-6095.

> **Tip**: Also refer to *z/VM General Information*, GC24-6095, for a list all of supported hardware that can be used with any release of z/VM.

The concept of having each user be a virtual machine is new to z/OS users. Therefore, we explain in the following sections how to manage them: start, resume, halt.

5.3.2 Starting a Guest Operating System

Each guest operating system is run from individual z/VM virtual machine userids. These userids are logged onto in the same manner as any other z/VM userid and usually comes up running in a CMS environment. From the log on session, a guest operating system can be loaded that virtual machines storage and run much the same as if it were running in an LPAR.

In z/VM, starting an operating system this way is still as “IPLing” (Initial Program Load). When you IPL an operating system, it takes control of the virtual machines storage and any other operating system that was running before is cleared from memory and no longer controls the virtual machine. Keep in mind that CP will still be running, because CP is not a guest operating system.

In CP, you perform an IPL by using the **IPL** command. CP can IPL directly from a device, in which case the operating system is read from the data residing on the device. Or CP can IPL from data stored in shared memory known as a *Named Saved System* (NSS). To IPL from a device, give the virtual device number of the device you wish to IPL; see Example 5-19 on page 250.

Example 5-19 IPL of the 190 disk

```
IPL 190
z/VM V5.3.0    2007-05-02 16:25
Ready; T=0.01/0.01 09:37:40
```

In Example 5-19 we IPLed, the 190 disk which by default contains the CMS operating system. At this point CMS is running and any command issued will be processed by CMS first.

An NSS is a copy of an operating system's kernel or nucleus, which has been saved in shared CP storage. Using an NSS to IPL an operating system has several advantages over using disks.

- Only one copy of the operating system will exist in memory no matter how many guests have IPLed it. This can lead to tremendous storage savings if you have numerous guests.
- Only one copy of the operating system exists, then updating everyone who uses it to a newer version is as simple as replacing that single NSS.

To IPL from an NSS, provide the name of the saved system; see Example 5-20. Most z/VM installations have a CMS NSS set up.

Example 5-20 IPL of the CMS named saved system

```
IPL CMS
z/VM V5.3.0    2007-05-02 16:25
Ready; T=0.01/0.01 09:37:40
```

Example 5-20 shows we IPLed CMS via the CMS NSS instead of the CMS disk. Notice that CMS starts exactly the same way that it did when we IPLed the 190 disk.

> **Note:** Some guest operating systems require that you perform a **SYSTEM CLEAR** command to clear out the guest's virtual memory before IPLing the guest. You may, alternatively, tell the **IPL** command to clear the memory for you by specifying the CLEAR parameter as shown here.
>
> `IPL 190 CLEAR`

5.3.3 Issuing CP commands while running a Guest Operating System

After a guest operating system is started, then all commands entered at the terminal will typically be processed by that operating system and not by CP. Sometimes, however, you may need to interact with CP for various reasons, such as linking to a new disk, finding out how many other users are on the system, changing your virtual networking hardware, or any other CP-related task.

CP provides a way for you to issue commands that bypass the guest operating system and go directly to CP. We use the **#CP command** for this purpose. You issue the command **#CP** followed by whatever CP command you want to execute.

For example, assume you start your guest operating system which only detects one CPU, but you think that your virtual machine has three CPUs. Example 5-21 on page 251 shows how you can verify that your virtual machine does indeed have three CPUs.

Example 5-21 Issuing a command to CP from within a guest operating system

```
#CP QUERY VIRTUAL CPUS
CPU 00  ID  FF02991E20948000 (BASE) CP   CPUAFF ON
CPU 01  ID  FF02991E20948000 STOPPED CP   CPUAFF ON
CPU 02  ID  FF02991E20948000 STOPPED CP   CPUAFF ON
```

When you issue a **#CP** command, CP will temporarily suspend your guest operating system long enough to complete your command, and then it will resume it. To your guest operating system, it does not appear as though time has stopped or that it has stopped running. This process is generally very fast and seamless.

Notes:

- This method of interacting with CP from within your guest operating system works flawlessly for almost all guest operating systems. However, it fails when you are running z/VM in a z/VM guest. In this case, you will actually need to pause the z/VM that is running as your guest operating system before you can issue commands directly to the underlying CP. Refer to 5.3.4, “Pausing a Guest Operating System” on page 252, for more detailed information about this topic.
- CMS will assume that any commands you issue that it doesn’t recognize are CP commands and it will pass them directly to CP for you. This means that you do not need to use the **#CP** command to force CP to execute commands when you are in CMS. You may simply execute the CP command normally and CMS will pass the command along to CP for you.

5.3.4 Pausing a Guest Operating System

CP also has a command for pausing your guest operating system temporarily. You may want to do this if you need to issue many CP commands and do not want the guest operating system running or interfering with your work.

You may also want to do this if you are a programmer who is debugging a guest operating system. To stop the guest operating system and return control to CP, enter the **#CP STOP** command; see Example 5-22 on page 252.

Example 5-22 Pausing a guest operating system

```
#CP STOP
```

As you can see from Example 5-22, there is no output from the #CP STOP command. However, the end result is that the guest operating system will be stopped and you will be able to interact directly with CP until you specifically resume running the guest operating system.

This method of pausing your guest operating system works flawlessly for almost all guest operating systems. However, it fails when you are running z/VM in a z/VM guest. When you are running z/VM in a z/VM guest, you essentially have CP running as your guest operating system. Therefore, you have *two* instances of CP to deal with:

- The first level CP is the instance of CP that started when you logged on to the system and your virtual machine was created.
- The second level CP was created when you IPLed it from within your first level CP instance.

When you are running a second level CP, the **#CP** command will issue commands to your second level CP—and *not* to your first level CP, as you might expect.

Figure explains the scenario when you have a second level z/VM system.

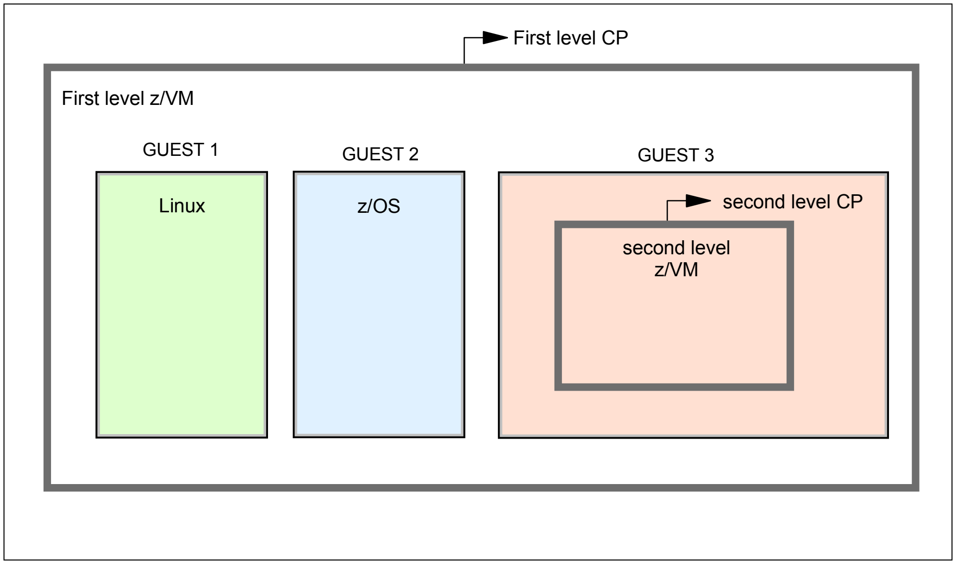

Figure 5-5 First and second level CPs.

To drop out of your second level CP and issue commands directly to your first level CP, press the PA1 key on the 3270 terminal keyboard. When you press the PA1 key (known as the Program Attention 1 key), then the second level CP (including any guest operating system it may be running) will be paused and your next command will be executed by your first level CP.

Note: When you press the PA1 key to pause a guest operating system, you may find that it only remains paused while you execute one command. After this it may resume again automatically. If this happens, it is because the RUN parameter is turned on for your first level virtual machine. The RUN parameter controls this behavior.

To turn off the RUN parameter after you have pressed the PA1 key and your first level CP is in control, execute the following command.

```
SET RUN OFF
```

From this point on you will be able to execute as many first level CP commands as you want and the second level CP will not resume until you tell it to.

You may also issue the **`SET RUN ON`** command to turn the RUN parameter on if you wish. Having RUN set to ON will ensure that your guest operating system remains running when you disconnect from the guest.

5.3.5 Resuming a Guest Operating System

You may resume your guest operating system (no matter what guest operating system you are running) with a simple **`BEGIN`** command. **`BEGIN`** takes no parameters and produces no output of its own, although you may see output from your guest operating system as it continues to execute.

Example 5-23 Restarting a stopped guest operating system

```
BEGIN
```

Note: In some cases you may be required to clear your 3270 terminal screen before your guest operating system will resume operation.

5.3.6 Halting a Guest Operating System

When you are finished with a guest operating system, use whatever command or procedure that operating system has for a proper shutdown, because not doing so can lead to data loss.

CP does, however, provide a command to halt the running guest operating system, clear the systems storage, and return directly to CP; see Example 5-24 on page 255.

Example 5-24 Halting a guest operating system

```
SYSTEM CLEAR
```

You may want to halt your guest operating system in this way if it has crashed or is stopped and is unresponsive to commands.

> **Note:** Because you will be running a guest operating system, you may need to precede the **SYSTEM CLEAR** command with **#CP**, as discussed earlier.

5.3.7 Managing CPUs

Processors are known as *CPU*s when they are virtual because CP reserves the term "processor" for real processors. A CPU is not actually a device because it does not have a virtual device number like all other devices do. Nevertheless, you can think of CPUs as devices because we manage them in a similar fashion.

Each virtual CPU has a *CPU address* which is like a virtual device number, but does not conflict with the virtual device number space. The virtual CPU address space is always represented in hexadecimal and ranges from *00* to *3F*.

You can define up to 64 virtual CPUs per virtual machine. It is also interesting to note that you can have a greater number of virtual CPUs defined than the system has real processors. Virtual CPUs are scheduled to run on real processors when real processor time is available.

Having more CPUs than processors is like running more programs than you have processors on your desktop workstation. CP handles this intricate scheduling of resources much like your desktop operating system handles scheduling for your programs and their processes.

> **z/OS analogy:** Managing logical CPUs in z/OS is done through the **CF CPU(n),ONLINE** or **CF CPU(n),OFFLINE** system commands.

Querying CPUS

Your virtual machine must have at least *one* CPU, but it may have more. You can obtain information about the CPUs on your virtual machine by using the **Query VIRTUAL CPUS** command (do not confuse this with the **Query CPU** command, which displays the virtual CPUid).

Example 5-25 Output from the QUERY VITUAL CPUS command

```
QUERY VIRTUAL CPUS
```

```
CPU 00  ID  FF02991E20948000 (BASE) CP   CPUAFF ON
```

This output contains one line per CPU. The virtual machine in this example only has one CPU, and that CPU address is 00. Its CPUID is *FF02991E20948000*.

Encoded in the CPUID is information about the hardware and the environment your virtual machine is running in. The details are not critically important and thus not covered here. Setting CPUAFF on means that CPU affinity is on for this CPU. “CPU affinity” refers to how having different types processors and CPUs is handled on your system; this is a complex subject and is beyond the scope of this textbook.

Notice the text (BASE); this denotes the *base CPU*. This means that this CPU was the first one created when you logged on to your virtual machine. This CPU is required by your virtual machine and cannot be detached or redefined. All other CPUs can be detached and redefined.

Defining CPUs

If you have the appropriate privileges, you can define more CPUs for your virtual machine. The ability to define more CPUS is defined in your virtual machine's directory entry; this is controlled by the system administrator.

If you are allowed to have more than one CPU defined, then you are given a limit somewhere between 2 and 64. You may define a new CPU by using the **DEFINE CPU** command; see Example 5-26.

Example 5-26 Defining a single virtual CPU

```
DEFINE CPU 1
CPU 01 defined
```

DEFINE CPU takes an argument that is either the CPU address you want the new CPU to have, or a dash-separated range of addresses if you wish to define more than one CPU; see Example 5-27.

Example 5-27 Defining multiple virtual CPUs

```
DEFINE CPU 1-3
CPU 01 defined
CPU 02 defined
CPU 03 defined
```

Note: CP does not provide a command for checking the maximum number of CPUs you are allowed to define. You need to determine this by either trial and error, or by checking your user directory entry for the value in the MACHine statement.

Detaching CPUs

If you decide that you do not need a virtual CPU that is already defined, then you may remove it by using the **DETach CPU** command. **DETach CPU** takes the same argument format as **DEFINE CPU**, which means you can detach a single CPU or a whole range of them with a single command; see Example 5-28.

Example 5-28 Detaching a CPU with the DETACH command

```
DETACH CPU 1
CPU 01 detached
```

Note: Detaching a CPU will cause your virtual machine to be reset and any running operating systems will be halted immediately—so use this command carefully.

5.3.8 Managing Storage (main memory)

Every virtual machine has memory. How much storage any particular virtual machine is allowed to have is determined by the system administrator and is defined in the user directory. “Storage” is not really a device, because it does not have a device number like all other devices do. Managing storage is simple because there is very little you need to know about it.

Your virtual machine’s memory is virtual, just like all of its other resources. Because it is virtual, it may exist anywhere in physical memory, or even on a disk (backing physical memory with disk-based storage is commonly referred to as “paging” or “swapping”).

As with CPUs, CP will allow you to overcommit real storage. When a guest is not logged on, it is not consuming any memory at all. It is only when a guest logs on that CP allocates some memory for it and its virtual CPUs and devices are created.

But even when a guest with 1 GB of storage logs on, it is not immediately using all 1 GB of its memory. If it is not running an operating system, then it is actually using an extremely small portion of its memory. CP only allocates physical memory to back a guest’s virtual memory when that guest actually uses the memory.

> **z/OS analogy:** In z/OS, managing storage would be similar to specifying the `REGION` for users, batch jobs, or STCs.

How much memory does your VM have

Determining how much memory you have is easy if you use CP's **`Query VIRTUAL STORAGE`** command; see Example 5-29.

Example 5-29 Output from the QUERY VIRTUAL STORAGE command

```
QUERY VIRTUAL STORAGE
STORAGE = 32M
```

In Example 5-29, the virtual machine has access to 32 Megabytes of storage. If you need more storage than is initially provided for you by CP, you might have the capability to increase your storage allotment.

When an administrator sets up a directory entry for a virtual machine, the administrator specifies two values for that virtual machine's storage. The first value is the initial amount of storage that virtual machine is given at logon time, and the second value is the maximum amount of storage that virtual machine is allowed to use.

Changing your storage size

You can change the amount of memory of your virtual machine by using the **`DEFINE STORAGE`** command. This command can be used to either reduce or increase storage capacity. It takes a single argument, which is the amount of storage that you would like your virtual machine to have.

The argument has two parts: a number, and then the unit of measure. Valid units of measure include those shown in Table 5-2.

Table 5-2 Valid units of storage measurement

Abbreviation	Amount of memory
K	Kilobytes
M	Megabytes
G	Gigabytes
T	Terabytes

Use this command carefully, because it causes a system reset which means it *clears* all of the virtual machine's memory and *stops* any running guest operating systems.

If you are simply running CP and not running a guest operating system, however, then there is nothing to worry about. In Example 5-30, we redefine our virtual machine to have 4 MB of storage.

Example 5-30 Setting the virtual machine's storage size to 4 megabytes

```
DEFINE STORAGE 4M
STORAGE = 4M
Storage cleared - system reset.
```

As shown in Example 5-31, we revert to our original size.

Example 5-31 Setting the virtual machine's storage size to 32 megabytes

```
DEFINE STORAGE 32M
STORAGE = 32M
Storage cleared - system reset.
```

Note: Your storage needs will vary depending on what your are doing with your virtual machine but chances are you do not need as much storage as you probably initially think. With a typical desktop computer we tend to estimate memory needs very high as memory grows increasingly cheaper and a high estimate imposes no performance penalty to the system. With z/VM it is important to understand your guest operating system, the workloads that will be run on it and their memory consumption characteristics.

Note: CP does not provide a command for checking the maximum storage size you are allowed to define. You can, however, determine this by simply attempting to define a storage size of 99 Terabytes (**DEFINE STORAGE 99T**) which will more than likely be far more than you are allowed to define. In this case, the resulting error message will tell you your maximum storage size.

5.3.9 Managing DASD

DASD is an acronym for *Direct Access Storage Device*. A DASD device is a simple magnetic disk that you have access to store data much the same as it is used in z/OS. The z/VM disks are allocated on portions of DASD volumes and contains files important to guest operating system.

Terminology

Before discussing DASD, it is important to understand some of the associated terminology and how z/VM manages it.

The mainframe that your z/VM instance is running on is connected to some type of physical disk that we will refer to as "real DASD". This DASD is segmented into units called DASD packs (also referred to as "volumes"). z/VM general users do not have the capability to examine the real DASD packs on the system.

Note: Real DASD packs can be varying sizes. The names and sizes for the most widely used are as follows:

- 3390-3 (also known as mod 3) 3339 cylinders - roughly 3 GB in size
- 3390-9 (also known as mod 9) 10017 cylinders - roughly 9 GB in size
- 3390-27 (also known as mod 27) 30051 cylinders - roughly 27 GB in size

The system administrator can dedicate a DASD pack to a guest. This means that access to the entire pack is given directly to that guest. This is known as a *dedicated DASD pack*.

A real DASD pack can contain many smaller virtual DASD packs (much as modern hard disks can be partitioned into many partitions that each appear to be a single disk). One of these smaller partitions is called a *minidisk*. Any number of minidisks can be created from a DASD pack, and each one can be arbitrarily assigned to a different guest. A guest can have many minidisks.

z/OS hint: Disk extents in z/OS are analogous to z/VM's minidisks.

A guest uses a minidisk and a dedicated DASD device in exactly the same ways. In both cases, the guest sees a virtual DASD device. A virtual DASD device is sometimes just referred to as a *disk*.

There are three different types of DASD to discuss here:

- Real DASD lives on a real physical disk, and is used to create minidisks.
- TDISK-based DASD lives on real DASD, but is only temporary and is destroyed when you log off.
- VDISK-based DASD lives in storage and is also destroyed when you log off.

For an in-depth discussion on other DASD operations, consult the *z/VM System Operation* guide[5].

Examining your DASD

To obtain a list of all of the virtual DASD devices that are available to you, use the **Query VIRTUAL DASD** command; see Example 5-32.

Example 5-32 Output from the QUERY VIRTUAL DASD command

```
QUERY VIRTUAL DASD
DASD 0190 3390 LX6RES R/O 107 CYL ON DASD  CD31 SUBCHANNEL = 0005
DASD 0191 3390 DKCD37 R/W 005 CYL ON DASD  CD37 SUBCHANNEL = 0000
DASD 019D 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0006
DASD 019E 3390 LX6W01 R/O 250 CYL ON DASD  CD32 SUBCHANNEL = 0007
DASD 0401 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0009
DASD 0402 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0008
DASD 0405 3390 LX6W01 R/O 156 CYL ON DASD  CD32 SUBCHANNEL = 000A
```

Example 5-32 on page 261 displays many different DASD devices available to us. The output shows one DASD device per line; look at each field in the first line in the example.

- The number immediately following the word DASD is the virtual device number of this DASD device (0190, in this case).
- The number immediately following that (3390, in all cases in this example) is the type of actual DASD hardware in use.
- The volume label (LX6RES) of the real DASD pack label that this virtual DASD device lives.
- The level of access (R/O) you have to the device.
- The size of this DASD device (107) is shown in cylinders (CYL).
- The real address of the DASD pack that this virtual DASD device resides on is displayed next (ON DASD CD31).
- Finally, the subchannel number for our virtual device is listed (SUBCHANNEL = 0005). A discussion of subchannels is beyond the scope of this book.

Here we explain the most significant parameters in more detail.

Virtual device number

As previously mentioned, the virtual device number uniquely identifies this device to your virtual machine. This can be any hexadecimal number that is not already defined to your virtual machine.

Type

The type used to be important many years ago when there were multiple types of real DASD, but except for the oldest installations, you should not run across any type of DASD today other than 3390. The type parameter is kept in place in order to maintain backward compatibility. One exception to this is when you are dealing

5 http://publibz.boulder.ibm.com/epubs/pdf/hcsf2b20.pdf

with VDISK (see “Virtual DASD (VDISK)” on page 265 for more information about this topic).

Volume label and real device address

The volume label and the real device address both apply to the real DASD pack that houses this particular virtual DASD device. These parameters are provided for informational purposes and you are not likely to need to refer to them unless you are a system administrator.

Size in cylinders

The size of DASD devices is given in cylinders. For 3390 DASD, a single cylinder is exactly equivalent to 849,960 bytes—which in turn is roughly equivalent to 850 KB.

Note:

- To determine how many megabytes your x cylinder DASD device holds, multiply x by 0.85.
- To determine how many cylinders you need for x MB of data, multiply x by 1.2.

Creating DASD

At some point, you may find that you need more disk space. However, the definition of a new DASD pack must be performed by an administrator, because a new virtual DASD device must be backed by a real disk somewhere, and general users do not have the authority to allocate from real disks.

You may, however, have the authority to create a TDISK or a VDISK; refer to “Temporary DASD (TDISK)” on page 263 and “Virtual DASD (VDISK)” on page 265s for details about these topics. Another option might be to ask your administrator for shared file pool (SFS) space; refer to 5.3.11, “The CMS Shared File System” on page 272 for further details.

Accessing other users’ DASD

You have learned the concepts behind accessing someone else’s disks in “Access another user’s disks (linking)” on page 136. With the proper authority you could read from and execute programs or even write to the other disk. Refer to that section for detailed information on the subject.

Removing DASD

You can remove a DASD device just like you would remove any other virtual device, by using the **`DETach`** command. `DETACH` takes the virtual device number of the device to detach as its only parameter.

Example 5-33 Example of the DETACH command

```
DETACH 592
DASD 0592 DETACHED
```

Do not be concerned that you will lose anyone else's data by detaching a disk that you linked. When you detach any disk, all you are doing is removing the virtual device within *your* virtual machine that points to the real disk. (This is not true, however, for TDISKs and VDisks.)

Temporary DASD (TDISK)

A TDISK is a virtual DASD device that is allocated from a pool of real DASD packs set aside specifically for the creation of temporary disks. TDISK is an abbreviation of temporary disk.

It is important to note that when you log off or the system fails, any TDISKs that you have created are destroyed and the data lost. A TDISK is meant to be used as temporary storage space.

Important: Do not store the only copy of important data on a TDISK.

Note: TDISKs can not be linked by other userids.

z/OS hint: A temporary z/VM DASD is similar to z/OS temporary data sets, for instance by specifying `DSN=&&DATASET,DISP=(NEW,DELETE)....)`.

Creating a TDisk

Not all z/VM installations will allow the creation of TDISKs. This might be the case for your installation if the system administrator has not set up the system for TDISK allocation, or if all TDISK space is in use at the time of your request.

However, you can request a TDISK allocation by using the **`DEFINE T3390 command`**. The T3390 argument specifies that you want to create a temporary model 3390 DASD device.

Note that **`DEFINE T3390`** takes two parameters: the virtual device number you want to assign to the new disk, and the size of this disk in cylinders. So, if you want a TDISK at device number *9FF* that is 100 cylinders in size, use the command shown in Example 5-34.

Example 5-34 Defining a TDISK

```
DEFINE T3390 9FF 100
```

```
DASD 09FF DEFINED
```

Note: If you want an *x* megabyte TDISK, multiply *x* by 1.2 to determine how many cylinders to specify during creation.

You can use the **Query VIRTUAL DASD** command to verify that your 9FF disk exists, as shown in Example 5-35.

Example 5-35 Verifying that a TDISK was created

```
QUERY VIRTUAL DASD
DASD 0190 3390 LX6RES R/O 107 CYL ON DASD  CD31 SUBCHANNEL = 0005
DASD 0191 3390 DKCD37 R/W 005 CYL ON DASD  CD37 SUBCHANNEL = 0000
DASD 019D 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0006
DASD 019E 3390 LX6W01 R/O 250 CYL ON DASD  CD32 SUBCHANNEL = 0007
DASD 0401 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0009
DASD 0402 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0008
DASD 0405 3390 LX6W01 R/O 156 CYL ON DASD  CD32 SUBCHANNEL = 000A
DASD 0592 3390 LX6W01 R/O 070 CYL ON DASD  CD32 SUBCHANNEL = 000B
DASD 09FF 3390 (TEMP) R/W 100 CYL ON DASD  59D4 SUBCHANNEL = 000C
```

Notice that there is no volume label for the TDISK, because it is residing in TDISK, and not on a specific real DASD pack.

In order to use this TDISK space to store temporary files, you will need to CMS format the disk when you access it. We'll use the disk file mode of F for this example.

Example 5-36 Format TDISK

```
format 9ff f
DMSFOR603R FORMAT will erase all files on disk F(9FF). Do you wish to
continue?
Enter 1 (YES) or 0 (NO).
1
DMSFOR605R Enter disk label:
tmp9ff
DMSFOR733I Formatting disk F
DMSFOR732I 100 cylinders formatted on F(9FF)
Ready; T=0.01/0.05 10:44:37
```

Removing a TDISK

You may remove a TDISK just like you would remove most other virtual devices, by using the **DETach** command; see Example 5-37. **DETach** takes the virtual device number of the device to detach as its only parameter.

Example 5-37 Removing a TDISK

```
DETACH 9FF
DASD 09FF DETACHED
```

Virtual DASD (VDISK)

A VDISK is a virtual DASD pack that resides in the z/VM system's central storage instead of on a real DASD. This is exactly like the concept of a RAM disk that you might be familiar with from other operating systems. Because they exist in storage, VDISKs tend to be significantly faster than normal DASD devices, and also significantly smaller since it takes central storage away from system use.

It is important to note that a VDISK exists exclusively in storage and is *not* backed by a real disk. This means that when you log off or the system fails, your VDISK is destroyed and the data is lost. A VDISK is meant to be a temporary storage space.

Important: Do not store the only copy of important data on a VDISK.

z/OS hint: VDISKs are similar to using VIO type unit definitions in z/OS.

Creating a VDisk

Not all z/VM installations will allow the creation of VDISKs. This might be the case for your installation if the system administrator has not set up the system for VDISK allocation, or if free memory is too scarce.

However, you can request a VDISK allocation by using the **`DEFINE VFB-512`** command. The VFB-512 portion is an acronym for Virtual Fixed Blocks of 512 bytes.

Note that the **`DEFINE VFB-512`** command takes two parameters: the virtual device number you want to assign to the new disk, and the size of this disk in 512-byte blocks. So, if you want a VDISK at device number *8FF* that is 16 MB in size, use the command shown in Example 5-38 on page 265.

Example 5-38 Creating a VDISK

```
DEFINE VFB-512 8FF 31250
DASD 08FF DEFINED
```

Note: If you want an *x* megabyte VDISK, multiply *x* by 2000 to determine how many blocks to specify during creation.

You can use the **Query VIRTUAL DASD** command to verify that your *8FF* disk exists, as shown in Example 5-39.

Example 5-39 Verifying that a VDISK was created

```
QUERY VIRTUAL DASD

DASD 0190 3390 LX6RES R/O 107 CYL ON DASD  CD31 SUBCHANNEL = 0005
DASD 0191 3390 DKCD37 R/W 005 CYL ON DASD  CD37 SUBCHANNEL = 0000
DASD 019D 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0006
DASD 019E 3390 LX6W01 R/O 250 CYL ON DASD  CD32 SUBCHANNEL = 0007
DASD 0401 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0009
DASD 0402 3390 LX6W01 R/O 146 CYL ON DASD  CD32 SUBCHANNEL = 0008
DASD 0405 3390 LX6W01 R/O 156 CYL ON DASD  CD32 SUBCHANNEL = 000A
DASD 0592 3390 LX6W01 R/O 070 CYL ON DASD  CD32 SUBCHANNEL = 000B
DASD 08FF 9336 (VDSK) R/W 31256 BLK ON DASD  VDSK SUBCHANNEL = 000C
```

Notice that in this example, the DASD type is 9336 for the VDISK, and not 3390 like it is for normal DASD. Also notice that there is no volume label or real DASD pack address listed for the VDISK, because it is not residing on a real disk. Also, the size is given in 512 byte blocks instead of cylinders.

You can see that the size of the disk that we received (31256 blocks) is slightly larger than the size we requested (31250). This is because VDISKs must be allocated in increments of 8 blocks. If you specify a size that is not an increment of 8, then CP will automatically round up for you so you get at least as much space as you asked for, and possibly more (but never less).

Another method of allocating VDISK space for a userid whenever it logs on is in the user directory. By adding the following statement in the MDISK area of a user's directory, VDISK space will be allocated each time the user logs on. Since this space is reallocated after each logon, any previous data destroyed.

Example 5-40 V-DISK statement in USER DIRECT

```
MDISK 9336 FB-512 V-DISK 31250   MR
```

Note: Other users will not be able to link to your VDISK unless it is defined in the user directory.

Also, if a VDISK is defined in your directory and you log off, it will ***not*** be destroyed if another user has linked to it.

Each allocation of V-DISK space will need to be CMS formatted before files can be stored on the disk.

Removing a VDISK

You can remove a VDISK just like you would remove any other virtual device, by using the **DETach** command. The **DETach** command takes the virtual device number of the device to detach as its only parameter; see Example 5-41.

Example 5-41 Removing a VDISK.

```
DETACH 8FF
DASD 08FF DETACHED
```

5.3.10 Spool devices

A *spool device* is used to process an ordered list of files or data kept in a queue. CP's spool devices are different from other i/o devices, in that they are not associated with real devices, such as DASD.

There are three different spool devices that you can have, each of which exists primarily to work with spool files. The specifics of what spool devices and files are and how they work can be quite complex, but most of the details are not very important for the vast majority of users.

Note: This book presents an extremely simplified view of spool devices, because most users may never need to know all of the specifics. For more information about this topic, however, you can refer to *Virtual Machine Operation*, SC24-6128.

Table 5-3 Types of spool devices

Spool device	Description
Reader	A virtual punch card reader
Punch	A virtual punch card punch
Printer	A virtual printer

For the purpose of our discussion, these devices can all be viewed as being similar in nature. More specifically, the commands that we use for managing files within the spool device queues can be used on all of them in the same manner. In our examples, we will concentrate on the reader device type.

z/OS hint: Spooling tasks in z/OS are performed within the Job Entry Subsystem (JES2 or JES3). Spool data is kept in `JES2 HASPACE` data set or in the JES3 equivalent.

Uses for spool devices

In the early days of z/VM, most installations would include real printers, punch card readers and punches. Back then these virtual devices would be used to interface to the real hardware, so you could actually work with the real devices. Today, however, most installations do not have these real devices. However, spool devices are used for handling a virtual machine's input and output.

The reader is the virtual machine's input holding area and can be seen as a mail inbox. From that inbox, we can **CP TRANSFER** files to other uses or **CMS RECEIVE** the files onto a user's mdisk storage.

You may, by using the CMS command **SENDFILE**, send a file from one of your disks to other users. Those users are then free to save the file from their reader to one of their disks. The **SENDFILE** command actually uses your punch device to "punch" the file into the reader queue of the appropriate user.

The virtual card reader is also often used for booting other operating systems or stand alone utilities, including Linux installer RAM files. When the file at the top of the reader queue is one that contains a loadable operating system or stand alone utility, **IPL 00C** will begin the load process.

> **Note:** CMS expects to find your reader at virtual device number *000C*, your punch at *000D*, and your printer at *000E*. For this reason, those devices are typically always created with the given device numbers.

Querying a spool device

Querying a spool device gives you a great deal of information, and it can be done by using the **Query** command in a similar way to querying other virtual devices.

In Example 5-42 we query our reader, which is typically defined with the virtual device number 000C. (If your reader is not addressed as 000C, use the **Query VIRTUAL ALL** command to find your reader as discussed in "Your virtual machine's resources" on page 73.)

Example 5-42 Querying a spool device

```
QUERY VIRTUAL 000C
RDR  000C CL * NOCONT NOHOLD   EOF       READY
000C 2540        CLOSED    NOKEEP NORESCAN  SUBCHANNEL = 0002
```

Most of the parameters shown in the output deal with the spool device configuration or how your spool files are treated by the spool device.

Creating a spool device

You create a spool device by using the **DEFINE** command. In most cases, your reader will have already been defined for you. But if it has not been defined, give as an argument to the **DEFINE** command the type of device you are creating and the virtual device number, as shown in Example 5-43 on page 269.

Example 5-43 Defining a spool device

```
DEFINE READER 00C

RDR  000C DEFINED
```

You can specify PUNCH and PRINTER in place of READER, to create those types of spool devices instead.

Removing a spool device

You can remove a spool device by using the **DETach** command, just like any other virtual device. Specify the virtual device number as an argument; see Example 5-44.

Example 5-44 Removing a spool device

```
DETACH 100
RDR  0100 DETACHED
```

This will work on punch and printer devices, as well.

Spool files

Spool files can be seen as files that are queued up for further processing. They are either destined to a virtual machine (punch), awaiting a virtual machines action (reader) or awaiting to be printed (print). None can be edited and only reader files can be read using the **CP PEEK** command.

> **Note:** CP does not have editing capabilities. If you need to edit a file that is in CP spool, it would need to be transferred to the reader queue (if it were in the punch or print queue) and then CMS RECEIVEd onto disk.

When your z/VM system was set up, your system administrator set aside some disk space for spool files. Any files that are in one of your virtual spool device queues actually exist on this disk space set aside as system spool space.

Displaying files in your reader queue

CP allows you to display the information about the files in your reader queue. Use the **Query READER ALL** command for this purpose, as shown in Example 5-45.

Example 5-45 Output from the QUERY READER ALL command

```
QUERY READER ALL
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE TIME     NAME      TYPE
FRED   0008 A PUN 00000017 001 NONE 06/07 13:46:01 TUX1    NETLOG
FRED   0009 A PUN 00000008 001 NONE 06/07 13:49:44 PROFILE   EXEC
```

Example 5-45 shows that there are two files in the reader queue: a file named `TUX1 NETLOG` and a file named `PROFILE EXEC`. There is a significant amount of output shown in this example, but the most important parts are listed in Table 5-4.

Table 5-4 Output from QUERY READER ALL command

Column	Description
ORIGINID	This is the user name of the virtual machine that sent the file to your reader queue.
FILE	This is the unique number assigned to the file. We will use this with other commands.
DATE & TIME	The date and time that this file was placed in the reader queue.
NAME & TYPE	The file name and type given to this file by the creator.

You can query the files in your printer and punch queues by substituting PRINTER or PUNCH for READER in the QUERY command shown in Example 5-45.

Adding files to your reader

Files are added to your virtual reader when another user sends a file to you or if a program is set up to create output that is destined for your userid.

If you wish to have a CMS file added to your reader, you could simply do a **`SENDFILE`** of that file to your userid. If you wish to have a file in your print or punch queue added to your reader, you could do a transfer of the file.

Example 5-46 Transfer a file from your print queue to your reader

```
q prt all
OWNERID  FILE CLASS RECORDS  CPY HOLD DATE  TIME NAME TYPE DIST
LNXGUI   0029 T CON 00000017 001 NONE 06/09 11:51:06       LNXGUI
Ready; T=0.01/0.01 10:20:26

trans prt 29 * rdr
RDR FILE 0029 SENT FROM USER1 PRT WAS 0029 RECS 0017 CPY  001 T NOHOLD
NOKEEP
0000001 FILE  TRANSFERRED
```

```
Ready; T=0.01/0.01 10:21:26

q rdr all
ORIGINID FILE CLASS RECORDS  CPY HOLD DATE  TIME NAME TYPE DIST
LNXGUI   0029 T CON 00000017 001 NONE 06/09 11:51:06      LNXGUI
Ready; T=0.01/0.01 10:22:26
```

Deleting files in your reader

You can remove a file from your reader queue by using the **PURge** command. **PURge** can be told to clear a single file, multiple files, or even all of the files in your reader queue. To clear a single file with **PURge**, specify which queue to delete from and the number of the file within that queue to delete; see Example 5-47 on page 271.

Example 5-47 Deleting a single file from your reader

```
PURGE READER 8
0000001 FILE  PURGED
```

CP responds that it deleted one file. If you want to delete multiple files, specify all of the file numbers for the files you want to delete, separated by spaces; see Example 5-48.

Example 5-48 Deleting multiple files from your reader

```
PURGE READER 8 9 10
0000003 FILES PURGED
```

CP now responds that it has deleted all three of the files specified. To delete all of the files in your reader queue, specify the **ALL** parameter to **PURge**, as shown in Example 5-49.

Example 5-49 Deleting all files from your reader

```
PURGE READER ALL
0000004 FILES PURGED
```

You can delete the files in your printer and punch queues by substituting PRINTER or PUNCH for READER in the PURGE command shown in Example 5-49.

Moving a file in your reader to someone else's reader

The CMS **SENDFile** command provides a useful way to move a file from one of your disks to someone else's reader queue. If, instead, you want to send a file from your reader queue to someone else's reader queue, use the CP **TRANsfer** command, as shown here.

```
TRANSFER yourID READER fileNumber destID READER
```

Table 5-5 on page 272 lists the TRANSFER command parameters.

Table 5-5 TRANSFER commands parameters

Argument	Description
yourID	This is your user ID. If you are a system administrator, you may actually have permission to transfer files from anyone's reader queue by specifying their guest name here.
fileNumber	This is the number of the file in your reader queue you want to send.
destID	This is the user ID of the person that you want to receive the file you are sending.

As shown in Example 5-50, we transfer file number 13 from user TUX1 to user FRED.

Example 5-50 Moving a file from TUX1's reader queue to FRED's reader queue

```
TRANSFER TUX1 READER 13 FRED READER
RDR FILE 0013 SENT TO FRED RDR AS 0025 RECS 0008 CPY 001 A NOHOLD
NOKEEP
```

The output shown in Example 5-50 states that the file was sent to FRED, as we have requested. If someone is logged on to FRED's console, then they will see a similar message alerting them to the fact that someone has sent them a file.

The **TRANsfer** command works with PRINTER and PUNCH devices as well. Simply substitute PRINTER or PUNCH for READER in the example.

5.3.11 The CMS Shared File System

z/OS programmers may be familiar with the z/OS Unix System Services (z/OS UNIX, or USS), which allows z/OS to access UNIX files. The most noticeable difference between the z/OS UNIX filesystem and z/OS data sets is that the first is an hierarchical file system when accessed from within USS.

z/VM has a filesystem that allows you to view the system in a hierarchical fashion: the CMS Shared File System (SFS). z/OS programmers may be more accustomed to this type of hierarchical file system.

Shared File System (SFS) is an additional file system that is shipped with z/VM, and it offers several advantages over the normal CMS file system. Table 5-6 lists and compares these file systems.

Table 5-6 Compare and contrast Shared File System and normal CMS file system

Shared File System	CMS file system
Sharing at file level	Sharing at minidsk level
Minimum allocation is zero 4096 byte blocks.	Minimum allocation 1 cylinder
Hierarchical file system	Flat file system
Allocation can be changed dynamically	Bigger minidisk required
Recovery at file level	Recovery at minidsk level

SFS is essentially a collection of minidisk storage managed by a server.

As can be seen in Figure 5-6, there are three main areas of interest:

Control data This is where the definitions of the filepool and its users are kept.

Log data Logs are kept so that, in the event of interruption, files can be rolled back.

User data This where the user data exists and the server will allocate data from here to give to users.

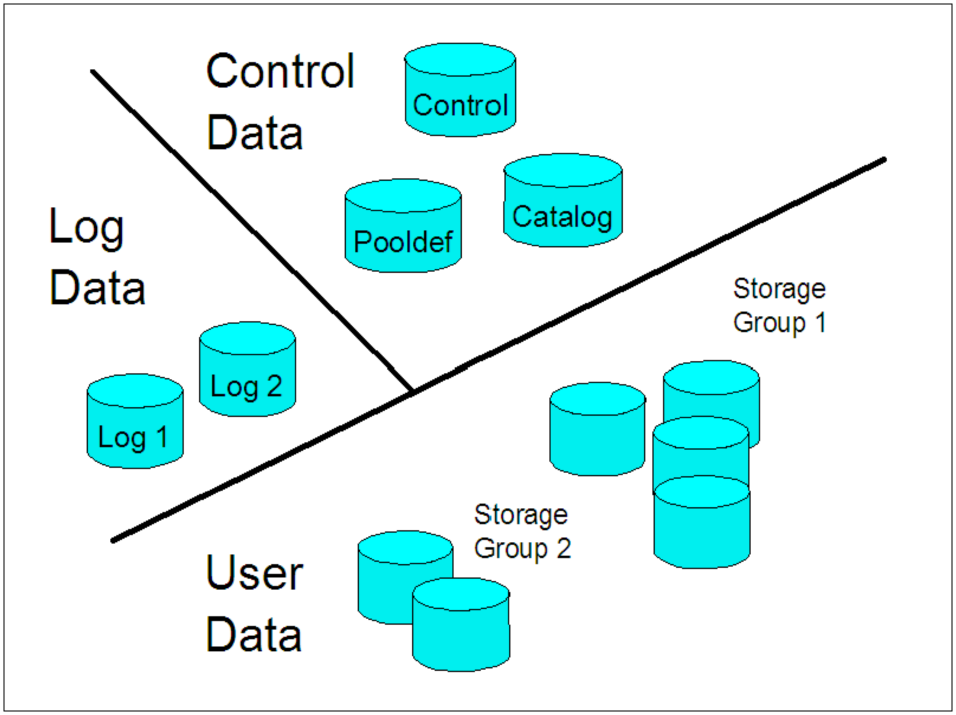

Figure 5-6 Shared File System

With SFS, instead of a minidisk, users are enrolled in a filepool by an administrator and will be given a number of 4096 byte blocks from a storage group that they can use in a similar way to a CMS minidisk.

Users will access the space in the same way that they do a CMS minidisk, but using the filepool name concatenated with their user ID instead of a minidsk device number. For example, they would use `VMSYSU:USERID` instead of 191 on and the **`ACCess`** command.

However, once accessed, users can use commands to create directories and subdirectories in a similar way as they do on a workstation, creating a hierarchical file structure if needed. Users can then GRANT authority at the file level to allow other users to read or write to data in its file space.

There are several commands, listed and described in Table 5-7, that users should be aware of if they have been allocated some space in the filepool.

Table 5-7 Commands related to SMS

Command	Description
SET FILEPOOL	Use the SET FILEPOOL command to set (or reset) your default file pool. `set filepool vmsysu`
QUERY FILEPOOL	Use the QUERY FILEPOOL command to display your current filepool. `query filepool current`
QUERY LIMITS	Use the QUERY LIMITS command to see how many 4 K blocks you have available to use. `query limits`
DIRLIST	Use the DIRLIST command to list you directories in a similar way that you use FILELIST to list CMS files.
ACCESS	Use ACCESS to assign a mode letter to your file space similar to the way that minidisks are accessed. `access vmsysu:cmsuser a`
CREATE	Use the CREATE command to create a directory similar to mkdir on a PC. `create directory test1`
GRANT	Use the GRANT command to grant other users access to files or directories.

The entire file system will have an administrator who is responsible for backups and allocation of minidisks to storage groups as required.

z/VM is shipped with three filepool servers:

- VMSERVR and VMSERVS are used by z/VM
- VMSERVU, a sample user file system

For more information about these topics, refer to *z/VM CMS File Pool Planning, Administration and Operation*, SC24-6074, and *z/VM CMS Commands and Utility Reference*, SC24-6073.

5.4 Job scheduling and batch jobs

z/VM and Linux offer ways of scheduling jobs to run at a particular time of the day/month/year and also to make them run periodically. This is useful to schedule non-disruptive system maintenance tasks, that is, no need to stop anything that any user may be doing at a given time in order to run those. Such tasks can be compacting the system logs to save space, update backups and so on.

The task of scheduling batch jobs, however, is not a common concept in z/VM or Linux. In fact, to run many “batch” processes you can either create a script for that or run all of them at once, knowing that they would eventually compete for I/O resources.

The following section gives an overview of tools similar to z/OS Job Entry subsystem (JES). You can simulate a batch processing queue using those scheduling tools and creating scripts.

5.4.1 Job scheduling

Job scheduling in z/VM

Scheduling a job is usually done with tasks that do not require human intervention, such as manipulating logs and creating backups. Doing this in z/VM, however, is not an usual task since administrators would usually use whichever tools are available in guest systems to perform those.

For the sake of completeness, the authors searched the web and found products, such as *CA VM:Schedule for z/VM*, that accomplish that.

Job Scheduling in Linux

Linux systems have a utility tool called *cron* to schedule programs to run periodically. Scheduling information is placed in the /etc/crontab file, using the following format:

```
minute hour day month year username command
```

You can specify each time component as an integer number. For example, use 1 through 12 for the months of January through December. Or you can specify one or more components as “*” characters, which will be treated as wildcards. For example, * in the month component means the command will run at the given day and time in every month. The username parameter should be the name of the user that executes the command, which might be different for security or authority reasons. For example, a backup process must be run by a user that can read all files in all directories. There is also the possibility for each user to put a crontab file in their home directory. For more information, refer to http://www.linuxdoc.org

Or you can run either of the following command, which will also provide more information:

```
man crontab
```

```
man 5 crontab
```

However, you don't have to use this text file; there is a simpler approach. Several directories exist in /etc where you can place a script file that will automatically, executed on specific moments in time. These directories are called:

- /etc/cron.hourly
- /etc/cron.daily
- /etc/cron.weekly
- /etc/cron.monthly

The script is then executed every hour, day, week, or month.

5.4.2 Running batch jobs

The term batch job originated in the days when punched cards contained the directions for a computer to follow when running one or more programs. Multiple card decks representing multiple jobs would often be stacked on top of one another in the hopper of a card reader, and be run in batches.

Today, jobs that run without any real-time dependencies or can be scheduled as resources permit, are called batch jobs. A program that reads a file and generates a report is considered to be a batch job.

z/OS uses a *job entry subsystem* or JES to manage the flow of work in the system. In the next sections you are presented with tools and concepts to simulate what z/OS users can do with JES.

Running batch jobs in z/VM: CMS Batch Facility

The *CMS batch facility* allows VM users to run their jobs in batch mode by sending jobs either from their virtual machines or through the real (system) card reader to a virtual machine dedicated to running batch jobs. The CMS batch facility then executes these jobs freeing user machines for other uses. If both the CMS batch facility and the Remote Spooling Communications Subsystem (RSCS) Networking Version 2 (or later version) are being executed under the same VM system, job input streams can be transmitted to the batch facility from remote stations by communication lines. Also, the output of the batch processing can be transmitted back to the remote station. The CMS batch facility virtual machine is generated and controlled on a user ID dedicated to execution of jobs in batch mode. The system operator generates the *batch machine* either by:

- Entering the `BATCH` parameter in the `PARM` field of the IPL command, or
- Specifying the `NOSPROF` parameter of the IPL command and entering the **`CMSBATCH`** command when the `VM READ status` appears.

After each job is executed, the batch facility IPLs itself, thereby providing a continuously processing batch machine. The batch processor IPLs itself by using the `PARM` option of the **`CP IPL`** command followed by a character string that CMS recognizes as peculiar to a batch virtual machine performing its IPL. Jobs are sent to the batch machine's virtual card reader from users' terminals and executed sequentially. When there are no jobs waiting for execution, the CMS batch facility remains in a wait state ready to execute a user job. The CMS batch facility is particularly useful for compute-bound jobs such as assemblies and compilations and for execution of large user programs, since interactive users can continue working at their terminals while their time-consuming jobs are run in another virtual machine. The system programmer controls the batch facility virtual machine environment by resetting the CMS batch facility machine's system limits, by writing routines that handle special installation input to the batch facility, and by writing exec procedures that make the CMS batch facility easier to use.

Note: For in-depth information on how to use the CMS Batch Facility, refer to the *CMS Planning and Administration* guide[a].

a. http://publibz.boulder.ibm.com/epubs/pdf/hcsg2b10.pdf

Simulating batch jobs in Linux

This section gives you a very simple overview on how to use scripts to simulate a batch-like processing. You can also schedule *cron* to run the scripts during the night to run, for example, daily sales reports.

The following example will make use of *shell scripting*, a language interpreted by many Linux shells[6], to control the batch behavior of the processes. This controlling script will run one program at a time. We use two simple programs that we wrote to sort the words of text files: one in natural order and the other in reverse other.

The script is shown in Example 5-51. It calls all programs found in the *./batch folder*, our sorting programs. Each of the programs has its corresponding input file in our the *./input* folder (where we placed the files for sorting) and each output is sent to our the *./output* folder. The only constraint to make our controlling script work is to name the input file after the same name of the batch program plus the suffix “-input”. Of course you can write better scripts than this.

Example 5-51

```
#!/bin/bash
```

[6] A shell is a program that interprets user inputs, calls the corresponding system's commands, and displays the output of that command. It is the user interface shown in Linux consoles after you log in.

```
for i in `ls ./batch`; do
    echo "Starting batch program $i"
    ./batch/$i < ./input/${i}-input > ./output/${i}-output
done
```

The input file to feed our first program, the one that sorts it in natural order, is shown in Figure 5-7 on page 279. The input for the second one is exactly the same, except that it says *this is unsorted file* **two** as opposed to *one*.

```
this
is
unsorted
file
one
d
g
t
e
w
a
i
o
p
```

Figure 5-7 Text file used as input.

To run the controller script, just do it as in Example 5-52 by calling the proper name of the script you created.

Example 5-52

```
ceron:~/scripts # ./controller.sh
Starting batch program 101_program
Starting batch program 102_program
ceron:~/scripts #
```

The outputs of our programs are our *./output* folder, and they are depicted in Figure 5-8 and Figure 5-9

```
ceron:~/scripts/output # cat 101_program-output
cat 101_program-output
Results of processing input data file : Succesfull completion at Fri
May 23 11:50:13 EDT 2008

a an d e file g i is o one p t this unsorted w
ceron:~/scripts/output #
```

Figure 5-8 Output of the batch program to sort files in natural order

```
ceron:~/scripts/output # cat 102_program-output
Results of processing input data file : Succesfull completion at Fri
May 23 11:50:15 EDT 2008

w unsorted two this t p o is i g file e d an a
ceron:~/scripts/output #
```

Figure 5-9 Output of the batch program to sort files in reverse order

Notice that each program created its output based on its input and accordingly to its processing. Note, also, that the timestamp differ by two seconds. We had coded our programs to send to the output the current system time upon completion of the job, and made them sleep for a couple seconds before returning to simulate a high load process. These two seconds show that the processes were run one after another, as our controlling script did it its *for* loop.

By using scripts similar to these and scheduling them to run in *cron* you can simulate z/OS batch jobs.

Have a look at http://en.wikipedia.org/wiki/Shell_script for pointers to shell scripting tutorials.

5.5 Cloning

As you can imagine, the process of repeatedly installing similar Linux images from scratch can quickly become tedious, so z/VM systems administrators need a simple, efficient, and repeatable process for creating these images on a z/VM system. Cloning a Linux image in z/VM takes matter of minutes before a new image is up. This can be helpful to quickly bring up a test or development environment or duplicate several production environments in matter of minutes than days. There is a lot of literature out there on this subject. The two resources are:

- z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES 10 (SG24-7493-00)
- Getting Started with Linux on System z version 5 release 3

5.6 System Monitoring

System monitoring is one of the most important activities to well administrate a computer system.Time will come in which you will need to look at the system's resource utilization in order to decide whether it should be given more storage, DASD or CPU. Alternatively, you will only be capable of getting the most out of your hardware if you can identify wastes of resource in an LPAR, Virtual Machine or Guest.

Tools exist in the z/VM and Linux worlds to allow you to accomplish the goals mentioned in the previous paragraph. Chapter 7, "Performance Monitoring and System Analysis" on page 321 gives a detailed overview on them.

5.7 Backing data up

As robust as a system may be, there is always a reason to back up your data from time to time:

- Natural disasters, such as a fire
- Unexpected power outage that causes data corruption
- Hardware failure
- Program errors that destroys data.

High end systems such as z/VM and Linux can not afford to loose data of a production system. Therefore, they have a variety of tools to help you back your data up. Chapter 8, "System events and logs, backup and recovery" on page 355 shows you a list of tools and commands available for that.

5.8 System Maintenance

Systems may need to be maintained by a number of reasons:

- Security updates

- Software level upgrade
- Installation of new features

For any of the previous reasons, it is not desirable to shut the production systems down in order to perform maintenance. In Chapter 9, “Applying system maintenance” on page 393 we talk about the ideal way to maintain a z/VM system and how it is compared to z/OS.

5.9 System debugging

As much as a system administrator can be careful about his system by continuously monitoring and maintaining it, problems do arise. Knowing where to look for information in order to fix them, however, is what makes the difference between an ordinary and a good administrator. Chapter 8, “System events and logs, backup and recovery” on page 355 explains which tools are available in z/VM and Linux that allows you to come to logical conclusions about a problem based on facts.

6

System administration

This chapter introduces some of the new and powerful features in z/VM on System z to help you perform some important administration tasks. You will learn about procedures and concepts related to System z security, user administration and privileges, and system networking.

Objectives: On completion of this chapter, you should be able to:

- Gain an understanding of some of the tasks involved in managing z/VM and Linux on System z security
- Learn some security related commands and statements
- Understand user classes
- Learn how to define users in z/VM

6.1 Manage z/VM Security

When deploying virtualization technologies for server consolidation you must never overlook security issues. Such issues, if not addressed with proper attention, can expose companies to risk. Inevitably, new technologies are frequently the target of new security threats.

In this chapter, we explain how z/VM provides security to your virtualized instances and provide an overview on how to secure your systems.

Within z/VM, the term security is a reference to the authentication and authorization schemes used to identify users and to control access to resources. System integrity, on the other hand, allows the z/VM Control Program (CP) to operate without interference or harm, intentional or not, from the guest virtual machines, as well as protecting the guest virtual machines from interfering with each other.

6.1.1 User Authentication

User login to a z/VM system is achieved by starting a terminal session with z/VM (local or telnet) and then providing a z/VM user ID and its associated password. Local terminal sessions are by definition highly secure since the data does not travel over a network. Remote terminal (telnet) or file transfer (ftp) sessions which travel over internal or external IP networks can be made highly secure by configuring and using the z/VM Secure Sockets Layer (SSL) support. The processing required for SSL is delivered through an SSL server supplied with z/VM, supporting 128-bit encryption and decryption services. The z/VM SSL server is a Linux for zSeries application that must be installed separately. Once the user has supplied the user ID and password, the Control Program validates the information. If the user ID and password are valid, the login is permitted and the terminal session is connected to the virtual machine's virtual console.

If you think of z/VM as a virtual computer room full of virtual servers, then think of a virtual machine user ID as a "virtual cage" around the server. No one can enter the cage unless they possess the key: the virtual machine password. This is very different from the discrete environment, where access to a machine room automatically gives access to all servers in that room.

Because the server console is protected by a z/VM password, you can more safely eliminate the protections normally given to a server's console. Automation techniques are greatly simplified if the automation tools do not have to, for example, enter the root password of a Linux server in order to shut it down or reboot it. While at first glance this may seem to reduce system security, it actually improves security by not requiring the root password to be known by the automation software.

A special capability available with z/VM is "Logon By." This function enables the system administrator to define a shared virtual machine. When the user enters the shared user ID, the user also provides his or her own user ID and password. In this way an audit trail is maintained of who is actually logged into a shared user ID and the problems inherent in sharing passwords are avoided.

Remote access protocols such as rexec, ftp, and nfs, all require the client to authenticate using a z/VM user ID and password. At no time will z/VM trust the claims of an unauthenticated client. Once authenticated, the remote client has the same access rights as the user would have if he or she were logged into the system with a terminal session.

For network applications, z/VM provides a Kerberos server and the programming interfaces that permit programs to take advantage of Kerberos authentication and encryption facilities. It should be noted that the IBM-provided network application suite and the z/VM Control Program do not use Kerberos authentication. While anonymous access to specific resources or to a virtual machine can be allowed by z/VM, such access must be explicitly enabled by the z/VM system administrator.

6.1.2 User Authorization

Once logged into the z/VM system the virtual machine can access various types of resources within the z/VM system, including entire DASD volumes, minidisks, tape drives, network adapters, user files, system files, and so on. The security features of z/VM are designed so that a virtual machine can access only the resources specifically permitted to it.

Those permissions may be given by the system administrator so that when the virtual machine is started, it automatically receives access to a certain resource, even if that virtual machine would otherwise be unable to access that resource. Alternatively, permissions may be given dynamically by the system administrator or the owner of the resource.

Some resources are accessible based on privilege class, others require additional authorization. The security facilities provided by z/VM can be enhanced according to any special or specific requirements for the customer’s environment by the addition of an External Security Manager.

It should be noted that while privileged commands can be used to change a running z/VM system, the system administrator may choose to set system configuration options during system initialization. This is accomplished by updates to the Control Program system configuration file. Consequently, access to the system configuration file must be tightly controlled.

6.1.3 Intrusion Detection

As an element of z/VM intrusion detection capabilities, if a login is denied, the denial is tracked and a security journal entry is made when the number of denials exceeds an installation-defined maximum. When a second maximum is reached, logon to the user ID is disabled, an operator message is issued, and the terminal session is terminated.

Journaling is supported on z/VM. Virtual machine logons and linking to other virtual machine's minidisks are detected and recorded. Using the recorded information, you can identify attempts to log on to a virtual machine or to link to minidisks using invalid passwords.

The TCP/IP component of z/VM will detect and report a variety of network intrusions, including SYN flooding, "Smurf" attacks and the "Ping o' Death".

The z/VM Control Program defines and assigns virtual processors to the virtual machine. These virtual processors are matched to the physical or logical (if z/VM is running in a logical partition) processors available to the Control Program. If the operating system running in the virtual machine is capable of using multiple processors, it will dispatch its workload on its virtual processors as if it were running in a dedicated hardware environment. This capability can be extremely useful to test a guest operating system in a multiprocessor mode, even on a uniprocessor system.

6.1.4 Virtual Processors Security

The Control Program handles dispatching the virtual processors on the available real processors. A real processor can either be dedicated to a single virtual machine or shared among multiple virtual machines. Bear in mind that the Control Program only handles the processors it controls, so if z/VM is running in an LPAR, the logical processors may in fact be shared with other LPARs.

So, we have virtual machines dispatching their work on one or more virtual processors, which are mapped to one or more logical processors, which may be mapped yet again to one or more physical processors. But don't worry. There is no significant security risk if the virtual, logical, or physical processor configuration is changed, or if work is dispatched on different physical processors. The state of a processor is preserved for one virtual machine and restored for another by the z/VM Control Program just as PR/SM™ does for

LPARs. Therefore, no information can be passed from one virtual machine to another via residual data in processor registers.

6.1.5 z/VM privilege classes

z/VM is a system of privilege; a user either can have no privileges or can be assigned to one or more privilege classes. Each privilege class represents a subset of Control Program commands that the system permits the user to run. Each privilege class, sometimes called CP privilege class, is defined around a particular job or set of tasks, thereby creating an area outside of which the user cannot execute any commands. A user can be assigned to more than one CP privilege class. Users are unable to enter commands in privilege classes to which they are not assigned. Here is a summary of CP privilege classes, their associated users, tasks, and security implications:

Privilege class A

The primary system operator. The system operator is among the most powerful and privileged of all z/VM users. The system operator is responsible for the system's availability and its resources. The system operator also controls accounting, broadcasts messages, and sets performance parameters.

Privilege class B

The system resource operator. The system resource operator controls the allocation and de-allocation of real resources, such as memory, printers, and DASD. Note that the system resource operator does not control any resource already controlled by the system operator or the spooling operator.

Privilege class C

System programmer. The system programmer updates the functions of the z/VM system and can change real storage in the real machine.

Privilege class D

Spooling operator. The spooling operator controls spool files and real unit record devices, such as punches, readers, and printers.

Privilege class E

System analyst. The system analyst has access to real storage and examines dumps to make sure that the system is performing as efficiently and correctly as possible.

Privilege class F

IBM service representative. The representative of IBM who diagnoses and solves problems by examining and accessing real input and output devices and the data they handle.

Privilege class G

General user. This is the most prevalent and innocuous of the CP privilege classes. The commands that privilege class G users can enter effect only their own virtual machines.

Privilege class ANY

The commands in this privilege class are available to any user.

Privilege classes A, B, C, D, E, and F require individuals worthy of significant trust and whose activities require careful auditing. For example, users with privilege class B or C can modify an installation's system of CP privilege. Because this modification violates the Controlled Access Protection Profile (CAPP) security policy, system programmers and similarly privileged users must be trusted to not tamper with the system of CP privilege (and auditing must confirm this trust).

> **Note:** Red Hat Enterprise Linux 5 is the first Linux operating system to ship with native support for the functionality necessary to meet Common Criteria for Trusted Operating Systems. This includes all functionality to enable EAL 4+ certification under the following protection profiles: CAPP (Controlled Access Protection Profile), RBAC (Role Based Access Control), and LSPP (Labeled Security Protection Profile).
>
> For more information, see Security benefits of Red Hat Enterprise Linux 5 on IBIBM System z. Document available at
> http://www.redhat.com/f/pdf/rhel/security_rhel5.pdf

As another example, privilege class C users can enter the cp store host command that allows them to alter real storage. This command makes it possible for users to negate the CAPP classification.

Privilege class G users have no influence outside their own virtual machines. So, with the exception of access to storage objects, they have very little security relevance.

The ANY privilege class commands cannot violate the security policies of the system. This is because all commands in the ANY privilege class are auditable and subject to either Discretionary Access Control (DAC) or Mandatory Access Control (MAC). Therefore, class ANY users, together with class G users, cannot violate the security policy. In the Control Program, each level of privilege is discrete and not predicated on others. Furthermore, each privilege class (a subset of commands) is related to one or more function types (subsets of users).

6.1.6 System user IDs involved in security

There are some users that are defined as part of the installation process. The standard user IDs for the default system installation are:

- OPERATOR: System operator and high privilege user ID. Similar to root in Linux systems.
- MAINT: Used by a person for system administration and maintenance. Similar to OPERATOR from an authorization point of view.
- EREP: EREP stands for Environmental Record Editing and Printing Program. Hardware anomaly detection and predictive failure system.
- DISKACNT: Records events such as logon and logoff.
- OPERSYMP: Used to retrieve symptom records. System dump analyzer and problem tracking system.

6.1.7 Security relevant statements

The system configuration file can be edited to control the system security. The configuration file is located on a partition of a volume allocated as PARM. This minidisk is normally under user ID MAINT, and it is on minidisk address CF1. The file is called SYSTEM CONFIG by default, although its name can be altered during installation. The file is read at IPL time by the CP program that uses the statements contained in the file to configure the system.

The following list summarizes the statements that are contained in the configuration file that are relevant to security:

Define Command:

Use the command, **`define`**, to:

- Change the configuration of your virtual machine
- Change the configuration of your operating system
- Add a new alias for an existing CP command on your system
- Add a new CP command to your system
- Add a new version of an existing CP command to your system

- Add a new DIAGNOSE code to your system
- Add a new guest LAN to your system.

Disable Command:

Use this command to prevent CP from processing requests for the specified CP command during and after initialization.

Here are some useful Disable command parameters:

- DISABLE Device

This command prevents specific devices from accessing the host system.

- DISABLE Exits

This command prevents CP from calling all entry points and external symbols associated with one or more exit points.

- DISABLE HCD

Prevent HCM and HCD from controlling I/O configuration.

- Disable Diagnose

Use this command to prevent CP whether from processing requests for one or more locally-developed DIAGNOSE codes during and after initialization.

- Enable Diagnose

Use this command to permit CP to process requests for the specified CP command during and after initialization.

JOURNALing

Use the JOURNALING statement to tell CP whether to include the journaling facility, whether to enable the system being initialized to set and query the journaling facility, and what to do if someone tries to log on to the system or link to a disk without a valid password.

Modify Command

Use the MODIFY COMMAND or CMD statement to redefine an existing CP command on the system during initialization.

- Modify LAN: Use the MODIFY LAN statement to modify the attributes of an existing guest LAN during initialization.

- Modify Priv_Classes: Use MODIFY PRIV_CLASSES to change the privilege classes authorizing the following CP functions.
- MODIFY VSWITCH: Use the MODIFY VSWITCH statement to modify the attributes of an existing virtual switch.

Note: For the complete description of syntax and usage for the system configuration file, refer to z/VM CP Planning and Administration, SC24-6083

PRIV_CLASSES: Use the PRIV_CLASSES statement to change the privilege classes authorizing the following CP functions.

- SYSTEM_USERids: Use the SYSTEM_USERIDS statement to specify user IDs that will perform special functions during and after IPL. These functions include accumulating accounting records, system dump files, EREP records, and symptom records, and specifying the primary system operator's user ID and disconnect status.
- USER_DEFAULTS: Use the USER_DEFAULTS statement to define default attributes and permissions for all users on the system.

6.1.8 Resource Access Control Facility (RACF)

Resource Access Control Facility (RACF) licensed program can satisfy the preferences of the user without compromising any of the concerns raised by security personnel. The RACF approach to data security is to provide an access control mechanism that offers effective user verification, resource authorization, and logging capabilities. RACF supports the concept of user accountability. It is flexible, has little noticeable effect on the majority of end users, and little or no impact on an installation's current operation. RACF controls access to and protects resources on both multiple virtual storage (z/OS) and virtual machine systems. For a software access control mechanism to work effectively, it must be able to first identify the person who is trying to gain access to the system, and then verify that the user is really that person. With RACF, you are responsible for protecting the system resources, such as minidisks, terminals, and shared file system (SFS) files and directories, and for issuing the authorities by which those resources are made available to users.

RACF records your assignments in profiles stored in the RACF database. RACF then refers to the information in the profiles to decide if a user should be permitted to access a system resource. The ability to log information, such as attempted accesses to a resource, and to generate reports containing that information can prove useful to a resource owner, and is very important to a smoothly functioning security system. Because RACF can identify and verify a user's user ID and recognize which resources the user can access, RACF can record the events where user-resource interaction has been attempted. This function records actual access activities or variances from the expected use of the system.

RACF has a number of logging and reporting functions that allow a resource owner to identify users who attempt to access the resource. In addition, you or your auditor can use these functions to log all detected successful and unsuccessful attempts to access the RACF database and RACF-protected resources. Logging all access attempts allows you to detect possible security exposures or threats. The logging and reporting functions are:

- Logging: RACF writes audit records in a file for detected, unauthorized attempts to enter the system. Optionally, RACF can also writes records for authorized attempts or detected, unauthorized attempts to:
  - Access RACF-protected resources
  - Issue RACF commands
  - Modify profiles on the RACF database.
- Sending Messages: RACF sends messages to the security console for detected, unauthorized attempts to enter the system and for detected, unauthorized attempts to access RACF-protected resources or modify profiles on the RACF database.
- Keeping Statistical Information: Optionally, RACF can keep selected statistical information, such as the date, time, and number of times that a user enters the system and the number of times a single user accesses a specific resource. This information can help the installation analyze and control its computer operations more effectively. In addition, to allow the installation to track and maintain control over its users and resources, RACF provides commands that enable the installation to list the contents of the profiles in the RACF database.

Some features introduced with z/VM Version 5.3 and RACF Feature Level 5.3 include:

- Mixed-case 8-character passwords
- Mixed-case password phrases up to 100 characters, including blanks
- No longer possible to reset password to default group name
- Audit trail can be unloaded in XML format
- Remote authorization and audit through z/VM new LDAP server and utilities

> **Note:** For more information on RACF, **Chapter 2.6.16, "RACF Security Server for z/VM" on page 43**

6.2 Defining users

This section discusses the process of defining users or virtual machines in z/VM. It would mostly discuss this in comparison with defining a TSO/E user. That would be very similar to defining a STC user or even a batch user, also. It does not intend to discuss any topics regarding users of transaction systems and similar. It illustrates the process on a native z/VM system without the additional products DIRMAINT and RACF.

6.2.1 z/OS assumptions

This section outlines some z/OS aspects of defining users, to outline some of the activities involved in user definition and management on z/OS before discussing the same activities on z/VM. For more detailed information on maintaining the z/VM user directory, refer to:

z/VM: CP Planning and Administration, SC24-6083

z/VM:CP Commands and Utilities Reference, SC24-6081

Security products

In earlier days defining users to TSO would imply defining them in the **User Attribute Data Set** (**SYS1.UADS**) using the **ACCOUNT** command. By now, most installations have merged this data set into their **RACF** data base or similar for ISV products. When **Unix System Services** (**USS**) were introduced in **OS/390®**, it became almost impossible to run a z/OS system without having a security system like RACF or similar activated. As several of the basic system functions of z/OS, for instance TCP/IP, are using or exploiting USS, it is very difficult to operate without a security server of some sort. Even some of the installation material states that it is a prerequisite that the user responsible for installing z/OS should have access to different **resource classes** and similar. Without a security product, the standard modules or services performing these checks (SAF) would most certain fail these services.

The **UADS** (**or** RACF data base **or** both) contains data on every TSO user (the RACF base would also contain STC and batch idents and other security related data). The base contains the user id, the associated password, account number or code, the default procedure name and all attributes for each user. Most installations would probably implement some kind of grouping or role based security based on their business. For instance, all system programmers would typically be members of the same group defined to RACF to simplify management of access rules if such grouping has been implemented.

The base itself is stored in an encrypted format, so passwords and other data could not be retrieved from the base directly. This also applies to the ISV alternatives.

A typical process of defining users in z/OS

A typical set of tasks necessary to define a TSO, STC or batch user in z/OS would include establishing rules for the following areas:

Security product Users would have to be defined to the security server or product, such as RACF.

Storage management Rules for how to manage the user's data would have to be addressed. This could involve adjustment of SMS constructs, defining catalog aliases, preallocating ISPF profiles etc.

Other Some sites are also doing various tailoring for their TSO users, such as adjustment of TSO/ISPF JCL procedures or related services. Accounting might also be included. There could also be a need for performance related tasks, such as adjusting WLM policies.

For most z/OS sites, these tasks would be performed by different lines or departments within the organization.

6.2.2 Defining users in z/VM

This section discusses how users are defined to z/VM. For more details on how to do this, see Chapter 4., "Installing z/VM and creating Linux or z/OS guests" on page 127.

USER DIRECT file

The z/VM users would all have to be defined and maintained in the file USER DIRECT (provided DIRMAINT and RACF are not installed). This file exists in two forms; one or more **source** form on one or more CMS file(s), and in a compiled **object** form on a CP-formatted disk. Only one USER DIRECT could be active at any given instant, but multiple could be accessed. The USER DIRECT is read at IPL time in a predefined search order. First, the system residence volume is searched for a valid object entry. If no such entry exist, all CP-owned volumes are being searched, in the order in which they appear (in the CP-owned list). If CP can not find a valid object, z/VM will come up with a default user id of OPERATOR. By logging on to this user, you could then establish valid USER DIRECT object.

Source formats of USER DIRECT

The source version of USER DIRECT can exist in two formats, a monolithic one, where all statements are contained in one physical file. The other format is known as the cluster form. The latter would include an index file, which could point to one or several files containing the necessary statements to build an object USER DIRECT file.

Note: All references to USER DIRECT in this book is for the monolithic source format. For further details on the cluster format, refer z/VM: CP Planning and Administration

Defining MINIDISKs

User data in z/VM is divided into the minidisk entity. This implies dividing your disk volumes into physical extents with a starting cylinder address and counting the number of cylinders from that.

Important: This is very different from z/OS in that z/VM manages (or “allocates” in z/OS terms) user data in **physical** disk extents on the actual volser. These extents are called **minidisks** in z/VM terminology. Note that they *could* include entire disks !

6.2.3 USER DIRECT from a z/OS perspective

The USER DIRECT file contains all data and definitions necessary for a virtual machine to log on to the system. To put it simple, it contains all the data you would define in SYS1.UADS by using the ACCOUNT command.To some extent, it would also contain the similar data and definitions for data storage administration as in z/OS (see “A typical process of defining users in z/OS” on page 294). In addition, USER DIRECT also contains devices and mini disk definitions necessary for each user to log on successfully. This applies to CMS users, VM service machines as well as guest operating systems. The definition of devices and mini disks could be seen as the equivalent of a TSO user’s allocations established by means of the logon procedure JCL.

6.2.4 USER DIRECT definitions

Regardless of USER DIRECT source format (monolithic or cluster), definitions must be processed in the following order:

DIRECTORY Definition — Which consists of one or more DIRECTORY directory control statements that define the output object directories.

Global Definition — This section must begin with the GLOBALDEFS directory control statement. It also includes directory control statements which define global settings to be used across all user definitions.

PROFILE Definitions — Each of which begins with a PROFILE directory control statement and consolidates other directory control statements that are commonly used across multiple users.

USER Definitions — (Also called virtual machine definitions) each of which begins with a USER directory control statement and defines an individual virtual machine.

The control statements are placed under each of these definitions to make the USER DIRECTORY.

6.2.5 DISKMAP command

This command is used to map the disk extents of the z/VM disks. In this section we pay attention to the disk containing user minidisks, as we are going to define a new CMS user.

Important: Use of the DISKMAP command is optional, but it is highly recommended.

DISKMAP will report any gaps or overlaps on these disks. Gaps imply that you are not utilizing the disk fully, whereas overlaps means that users are using the same extent of the disk. The latter implies a potential risk of overwriting data, thus corrupting them, and should therefore be avoided.

The command itself is invoked by issuing:

DISKMAP *fn ft opt*

where ***fn*** is file name of the USER DIRECT source you want to examine, ***ft*** is the file type of it and ***opt*** could be specified as **(DOENDS** to cause MDISK statements that had "END" specified for the ending cylinder/block in the directory being mapped to be included in the DISKMAP output file.

Output from DISKMAP is stored in file **fn DISKMAP** (fn=filename of the mapped file) on the **A** drive of the user running it. In our example, we run the command **`diskmap user`**. Figure 6-1 on page 297 displays the output of file *user diskmap a* (note that some of the lines are removed from output due to an excessive number of lines). Notice that there is a gap on line 00329 starting at cylinder 1886 for 114 cylinders and ending at 1999. The user must have at least privilege class B to run this command.

```
 USER     DISKMAP  A1  F 80  Trunc=80 Size=393 Line=240 Col=1 Alt=0

00240 LX6W02   $ALLOC$     A03     3390       00000        00000        00001
00241          40SASF40    2D2     3390       00001        00150        00150
00242          OSADMIN2    191     3390       00151        00160        00010
00243          OSADMIN3    191     3390       00161        00170        00010
-----------------------SNIP-------------------------------------------
USER     DISKMAP  A1  F 80  Trunc=80 Size=393 Line=320 Col=1 Alt=0

00320          5VMHCD20    2D2     3390       01454        01553        00100
00321          5VMHCD20    29D     3390       01554        01565        00012
00322          5VMHCD20    300     3390       01566        01625        00060
00323          5VMHCD20    400     3390       01626        01685        00060
00324          CBDIODSP    191     3390       01686        01805        00120
00325          COSTA       191     3390       01806        01815        00010
00326          HAIMO       191     3390       01816        01825        00010
00327          LNXGUI     0191     3390       01826        01835        00010
00328          COSTA       192     3390       01836        01885        00050
00329                                          1886         1999          114
GAP
00330          GUILL       191     3390       02000        02009        00010
00331          OMAR        191     3390       02010        02019        00010
00332          RAY         191     3390       02020        02029        00010
00333          KEN         191     3390       02030        02039        00010
00334          CERON       191     3390       02040        02049        00010
00335          LNXMAINT    191     3390       02050        02069        00020
00336          LNXMAINT    192     3390       02070        02369        00300
00337          LYDIA       191     3390       02370        02379        00010
00338          TOSK        191     3390       02380        02389        00010
00339          ZOS1       0191     3390       02390        02391        00002
====>
```

Figure 6-1 Output from DISKMAP

So, based on the output from DISKMAP, minidisk(s) for the user to be defined would have to start in CYLINDER 2380 not to cause overlaps (see line 00337 for LYDIA 191 ending in cylinder 2379).

6.2.6 Creating and maintaining the USER DIRECT source file

This is done by creating or editing an existing CMS file using the z/VM editor XEDIT (see Chapter 2.8.3, "XEDIT" on page 72). The default name of this file is USER DIRECT residing on MAINT's virtual disk 2CC. Installations could use this as a primer, or write their own. When the source file is edited, for instance when a new user have been added, the source file is compiled to object format using the DIRECTXA command. Maintenance of the USER object (by means of command **DIRECTXA**) has to be performed from a virtual machine that has privilege class of A, B or C (or similar installation defined ones).

It is a good idea to perform a syntax check of the contents of the source file (as well as running the DISKMAP command (see Chapter 6.5.8, "DISKMAP command" on page 285) to check for possible disk overlaps).

The syntax for the DIRECTXA command is (see Figure 6-2 on page 287):

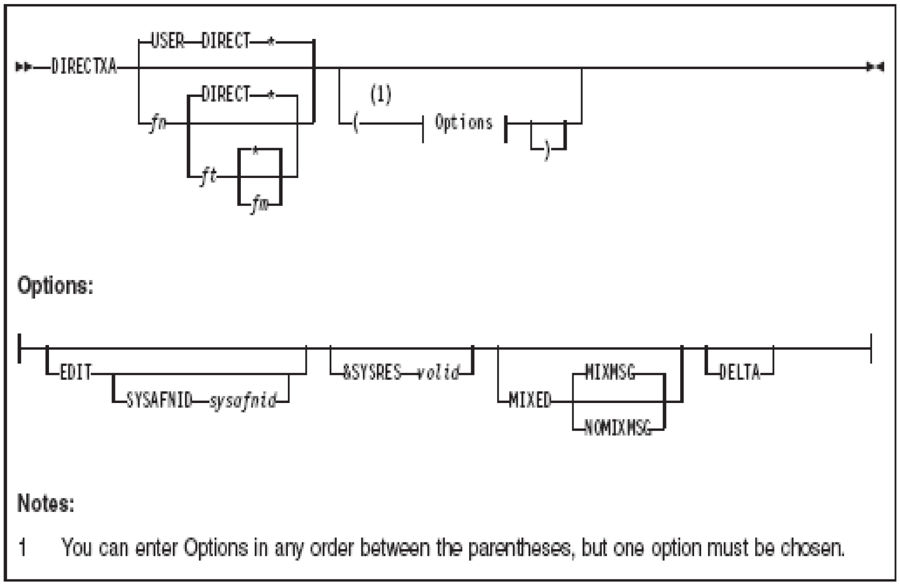

Figure 6-2 Syntax for DIRECTXA command

Note that filename (*fn)* and filetype (*ft)* have *default* values of **USER** and **DIRECT**, respectively. So commands:

DIRECTXA MYDIR DOCUMENT

and

DIRECTXA MYDIR (provided files **MYDIR DOCUMENT** and **MYDIR DIRECT** exist)

works just the same, if you prefer a different name for the source version of the file.

Syntax checking of the edited source file is performed issuing command:

DIRECTXA *fn ft* * (EDIT

This is a useful function, especially for personnel with limited experience in maintaining this type of file.

Example:

The source file (MYDIR DOCUMENT) contains the following statement, as shown in Figure 6-3:

```
PROFILE IBMDFLT
  SPOOL 000C 2540 READER *
  SPOOL 000D 2540 PUNCH A
  SPOOL 000E 1403 A
  CONSOLE 009 3215 T
  LUNCH MAINT 0190 0190 RR
  LINK MAINT 019D 019D RR
```

Figure 6-3 Sample MYDIR DOCUMENT

Instead of **LINK**, the administrator has punched **LUNCH**.

By issuing

DIRECTXA MYDIR DOCUMENT * (EDIT

the following response is issued from DIRECTXA (Figure 6-4 on page 300):

```
z/VM USER DIRECTORY CREATION PROGRAM – VERSION 5 RELEASE 3.0
   LUNCH MAINT 0190 0190 RR
HCPDIR751E INVALID OPERAND – LUNCH FOLLOWING PROFILE IBMDFLT
HCPDIR1775E PROFILE DEFINITION IBMDFLT IS INVALID AND WILL NOT BE PROCESSED FOR
ANY USER DEFINITIONS THAT INCLUDE IT.
EOJ DIRECTORY NOT UPDATED
```

Figure 6-4 Response from DIRECTXA

We now proceed into actually defining the new user, with a User ID of TOSK, also defining a corresponding 191 minidisk. This is the actual portion of USER DIRECT regarding user TOSK (see Figure 6-5 on page 300).

```
01967 *************************************************
01968 USER TOSK    TOSK       32M 64M G
01969    INCLUDE IBMDFLT
01970    ACCOUNT  ACT4  CMSTST
01971    MACH XA
01972          IPL 190
01973     MDISK 191 3390 2380 010 LX6W02  MR  READ    WRITE  MULTIPLE
```

Figure 6-5 Statements to define user TOSK

Notice in Figure 6-5 on page 300 the USER DIRECT statements:

Line 01968	Statement USER initiates the statements for the user. The first occurrence of TOSK is the userid, the second is password. 32M is the initial amount of storage defined to this user, 64M is the maximum amount of storage. The G is the privilege class(es) assigned to the user, in this example only a general user
Line 01969	Includes what is known or defined as a PROFILE in USER DIRECT, should be seen as statements common for several users
Line 01970	Assigns account code for the user
Line 01971	Sets the architecture for user TOSK
Line 01972	States that this user should be IPLed from 190, corresponds to CMS
Line 01973	Assigns the minidisk 191 on a 3390 device; starting at cylinder 2380 for 10 cylinders on volume LX6W02 in multiple read, establish passwords for READ (only), WRITE and MULTIPLE READ, respectively. These are used when other users access TOSK's 191.

Invoking the following command:

DIRECTXA USER DIRECT (EDIT

would now give the response shown in Figure 6-6 on page 301.

```
Ready: T=0.01/0.01 15:28:21
Directxa user direct (edit
z/VM USER DIRECTORY PROGRAM – VERSION 5 RELEASE 3.0
EOJ DIRECTORY NOT UPDATED
Ready: T=0.01/0.01 15:28:33
```

Figure 6-6 Response from DIRECTXA without errors

In order to check for disk gaps or overlaps, the following command is issued:

DISKMAP USER DIRECT (DOENDS.

The response is shown in Figure 6-7

```
Diskmap user direct (doends
File USER DISKMAP A has been created.
Ready: T=0.04/0.04 15:39:12
```

Figure 6-7 DISKMAP response

Browsing the USER DISKMAP file shows no gaps or overlaps within this disk's extents, with the exception that MAINT is overlapping the same extent. This is due to MAINT having this disk (and several others) defined as a minidisk from cylinder 0 to END. This is a normal way of defining CP-owned disks.

By issuing the command

DIRECTXA USER DIRECT

the updated source is compiled to an object and brought online to z/VM. User id TOSK can now log on.

6.3 Networking access

This section describes how to give network access to your guest operating systems. We discuss how you define that access for LANs and VSWITCHs.

Define LAN

Use this command to create a guest LAN which can be shared among virtual machines on the same VM system. Each guest LAN is identified by a unique combination of *owner id* and *lan name*. A VM user can create a simulated network interface card (NIC) and connect it to this LAN segment.

> **Note:** The SET LAN command can be used to modify attributes of the guest LAN

You can also define a LAN during system initialization using the DEFINE LAN configuration file statement. The Class B form of the command allows the invoker to create a LAN for another user (for example, OWNERid SYSTEM) and specify whether accounting is on or off for the LAN being defined.

Define VSWITCH:

Use this command to create a CP system-owned switch (a virtual switch) to which virtual machines can connect. Each switch is identified by a *switch name*. A z/VM user can create a simulated QDIO network interface card (NIC) and connect it to this switch with the NICDEF directory statement. Under the DEFINE VSWITCH statement, the VLAN parameter is important, but optional, if you want to isolate guests subnets based on VLAN IDs.

6.4 Linux on System z resource management

This section discusses the necessary steps to enable dynamically new resources - disks, network interface card, CPU - to a Linux guest.

Going into the details of Linux on System z hardware detection and configuration is beyond the scope of this book. Nevertheless, it is necessary to give a brief outline of the processes involved to understand the next section.

6.4.1 Hardware detection and configuration

Devices nodes and udev

The Linux kernel represents the character and block devices it knows as a pair of numbers, *<major>*:*<minor>*. Some major numbers are reserved for particular device drivers, others are dynamically assigned to a device driver when Linux boots. For example, major number 94 is always the major number for DASD devices while the device driver for channel-attached tape devices has no fixed major number. A major number can also be shared by multiple device drivers.

The device driver uses the minor number *<minor>* to distinguish individual physical or logical devices. For example, the DASD device driver assigns four minor numbers to each DASD: one to the DASD as a whole and the other three for up to three partitions. Device drivers assign device names to their devices, according to a device driver-specific naming scheme. Each device name is associated with a minor number.

For more information, please refer to *Devices drivers, Features and Commands, SC33-8289.* The latest version can be found on Developerworks website at hhttp://www.ibm.com/developerworks/linux/linux390/october2005_documentation.html

Historically, Linux provides a large set of pre-defined devices nodes in the /dev directory. Most of the time, only a small subset of these devices nodes were used. With Linux 2.6, things changed: there is only a small number of pre-defined devices nodes in /dev directory, and a user-space dameon, udev, in charge of creating the required device nodes in /dev dynamically when notified by the kernel that a new device is plugged in. Udev can also be used to “tune” the device nodes names to fit an organization’s needs or set permissions on nodes, for instance.

For more information about udev capabilities, the reader can refer to udev manual page (man 7 udev) or

/sys filesystem

Once a device has been brought online in Linux, devices drivers expose their parameters to the userspace through an in-memory file system called /sys.

> **Note:** /sys filesystem doesn’t exist on disk. It is recreated each time a Linux guest IPLs.
>
> /sys is the successor of /proc filesystem, that still exists for downward compatibility.

This allows initialization scripts or users to interact with the drivers parameters, to query or modify the configuration. For more information about the /sys filesystem structure, please refer to *Devices drivers, Features and Commands, SC33-8289.* The latest version can be found on Developerworks website at hhttp://www.ibm.com/developerworks/linux/linux390/october2005_documentation.html

6.4.2 Disk management

Enabling a disk in Linux

Once the new disk as been defined and in the directory, and dynamically attached to the guest (as explained in Chapter 5.3.9, "Managing DASD" on page 259), it is time to enable it in Linux.

Example 6.1 shows the output of the lscss command before and after attaching the disk as device 0200.

> **Note:** lscss commands stands for "List Channel Subsystem", and shows all the devices that are attached to you system.

Example 6-1 lscss command output, before and after attaching a new dasd to the guest

```
lnxguill:~ # lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.0191 0.0.0000  3390/0A 3990/E9      F0  F0  FF   52565A5E 00000000
0.0.0100 0.0.0001  3390/0C 3990/E9 yes  F0  F0  FF   5054585C 00000000
0.0.0101 0.0.0002  9336/10 6310/80 yes  80  80  FF   00000000 00000000
0.0.0150 0.0.0003  3390/0C 3990/E9 yes  F0  F0  FF   5054585C 00000000
[...]
lnxguill:~ # lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.0191 0.0.0000  3390/0A 3990/E9      F0  F0  FF   52565A5E 00000000
0.0.0100 0.0.0001  3390/0C 3990/E9 yes  F0  F0  FF   5054585C 00000000
0.0.0101 0.0.0002  9336/10 6310/80 yes  80  80  FF   00000000 00000000
0.0.0150 0.0.0003  3390/0C 3990/E9 yes  F0  F0  FF   5054585C 00000000
[...]
0.0.0200 0.0.0012  3390/0C 3990/E9      F0  F0  FF   5054585C 00000000
lnxguill:~ #
```

At this point, the disk is seen by Linux, but it has to be enabled for the operating system to be able to use it. The disk is enabled with chccwdev command, provided by IBM, and delivered in the s390-tools package, as shown in Example 6-2 on page 304.

Example 6-2 Enabling a disk for Linux

```
lnxguill:~ # chccwdev -e 0.0.0200
Setting device 0.0.0200 online
Done
```

```
lnxguill:~ #
```

The -e switch stands for Enable. To disable a device, the -d switch has to be used. When the disk is brought online, the Linux kernel assigns the disk a device node name, that will be used for referring to the disk. The device node name attributed to the disk can be found using the lsdasd command, as depicted below in Example 6-3.

Example 6-3 Defining the device node name

```
lnxguill:~ # lsdasd
0.0.0100(ECKD) at ( 94:  0) is dasda      : active at blocksize 4096, 126000
blocks, 492 MB
0.0.0101(FBA ) at ( 94:  4) is dasdb      : active at blocksize 512, 200000
blocks, 97 MB
0.0.0150(ECKD) at ( 94:  8) is dasdc      : active at blocksize 4096, 456840
blocks, 1784 MB
0.0.0200(ECKD) at ( 94: 12) is dasdd      : active at blocksize 4096, 1802880
blocks, 7042 MB
```

The device 0200 (as defined in z/VM DIRECTORY) will be accessed by Linux through node /dev/dasdd.

Note: It is also possible to add disks thanks to YaST graphical interface. Go to Hardware > DASD, then select the device you want to enable for Linux, and select Activate in the drop-down list.

Depending on the intended use of the disk device, it can be necessary to create a file system on it before using it.

Make changes permanent

The steps detailed above allows an administrator to dynamically add a disk device to a running Linux, without the need to reboot. But, at next IPL, changes will be lost.

To make changes permanent in SLES10, here are the necessary steps to follow

1. Create a configuration file for the device in /etc/sysconfig/hardware, using the template in /etc/sysconfig/hardware/skel/hwcfg-dasd-eckd, as shown in Example 6-4

   Example 6-4 Disk device configuration file

   ```
   lnxguill:/etc/sysconfig/hardware # cp -arv skel/hwcfg-dasd-eckd
   hwcfg-dasd-bus-ccw-0.0.0200
   ```

```
`skel/hwcfg-dasd-eckd' -> `hwcfg-dasd-bus-ccw-0.0.0200'
```

This file will be used at IPL to setup the dasd driver accordingly.

2. To make sure the Linux device nodes are created in the correct order, it is necessary to rebuild the initial ramdisk and rebuild the boot record each time a new DASD is added to Linux. See Example 6-5 for an example.

Example 6-5 Rebuild ramdisk, and update boot record

```
lnxguill:/etc/sysconfig/hardware # mkinitrd
Root device:    /dev/dasda1 (mounted on / as ext3)
Module list:    jbd ext3 dasd_eckd_mod dasd_fba_mod (xennet xenblk)

Kernel image:   /boot/image-2.6.16.21-0.8-default
Initrd image:   /boot/initrd-2.6.16.21-0.8-default
Shared libs:    lib64/ld-2.4.so lib64/libacl.so.1.1.0
lib64/libattr.so.1.1.0 lib64/libblkid.so.1.0 lib64/libc-2.4.so
lib64/libcom_err.so.2.1 lib64/libdl-2.4.so lib64/libext2fs.so.2.4
lib64/libhistory.so.5.1 lib64/libncurses.so.5.5
lib64/libpthread-2.4.so lib64/libreadline.so.5.1 lib64/librt-2.4.so
lib64/libuuid.so.1.2
Driver modules: dasd_mod dasd_eckd_mod dasd_fba_mod
DASDs:           0.0.0100(ECKD) 0.0.0101(FBA) 0.0.0150(ECKD)
0.0.0200(ECKD)
Filesystem modules:     jbd ext3
Including:      initramfs fsck.ext3
17062 blocks

initrd updated, zipl needs to update the IPL record before IPL!
lnxguill:/etc/sysconfig/hardware # zipl
Using config file '/etc/zipl.conf'
Building bootmap in '/boot/zipl'
Building menu 'menu'
Adding #1: IPL section 'ipl' (default)
Adding #2: IPL section 'failsafe'
Preparing boot device: dasda (0100).
Done.
lnxguill:/etc/sysconfig/hardware #
```

Note: For some device drivers, the assignment of minor numbers and names can change between kernel boots, when devices are added or removed in a VM environment, or even if devices are set offline and back online. The same file name, therefore, can lead to a completely different device.

That's why to ensure all your disks are brought online at IPL, in the order you activated them, we rebuild the initial ramdisk.

In RHEL5, the steps to make the changes permanent are:

1. Update the /etc/modprobe.conf file as shown in Example 6-6, adding the new dasd at the end of the options dasd_mod= line:

Example 6-6 Updating modprobe.conf

```
[root@mbase40 etc]# cat /etc/modprobe.conf
options dasd_mod dasd=0100,0150,0101,200
alias eth0 qeth
alias hsi0 qeth
```

2. Rebuild the initial ramdisk, and update the boot record, as detailed below in Example 6-7:

Example 6-7 Recreating the initial ramdisk and updating boot record in RHEL5

```
[root@mbase40 etc]# mkinitrd -v /boot/initrd-2.6.18-8.1.3.el5.img
2.6.18-8.1.3.el5
Creating initramfs
Looking for deps of module uhci-hcd
Looking for deps of module ohci-hcd
Looking for deps of module ehci-hcd
Looking for deps of module ext3: jbd
Looking for deps of module jbd
Looking for driver for device dasda1
Looking for deps of module ccw:t3990mE9dt3390dm0C: dasd_mod
dasd_eckd_mod
Looking for deps of module dasd_mod
Looking for deps of module dasd_eckd_mod: dasd_mod
Looking for driver for device dasdc1
Looking for deps of module ccw:t6310m80dt9336dm10: dasd_mod
dasd_fba_mod
Looking for deps of module dasd_fba_mod: dasd_mod
Looking for deps of module ide-disk
Using modules:
/lib/modules/2.6.18-8.1.3.el5/kernel/fs/jbd/jbd.ko
/lib/modules/2.6.18-8.1.3.el5/kernel/fs/ext3/ext3.ko
```

```
/lib/modules/2.6.18-8.1.3.el5/kernel/drivers/s390/block/dasd_mod.ko
/lib/modules/2.6.18-8.1.3.el5/kernel/drivers/s390/block/dasd_eckd_mo
d.ko
/lib/modules/2.6.18-8.1.3.el5/kernel/drivers/s390/block/dasd_fba_mod
.ko
/sbin/nash -> /tmp/initrd.s14762/bin/nash
/sbin/insmod.static -> /tmp/initrd.s14762/bin/insmod
copy from `/lib/modules/2.6.18-8.1.3.el5/kernel/fs/jbd/jbd.ko'
[elf64-s390] to `/tmp/initrd.s14762/lib/jbd.ko' [elf64-s390]
copy from `/lib/modules/2.6.18-8.1.3.el5/kernel/fs/ext3/ext3.ko'
[elf64-s390] to `/tmp/initrd.s14762/lib/ext3.ko' [elf64-s390]
copy from
`/lib/modules/2.6.18-8.1.3.el5/kernel/drivers/s390/block/dasd_mod.ko
' [elf64-s390] to `/tmp/initrd.s14762/lib/dasd_mod.ko' [elf64-s390]
copy from
`/lib/modules/2.6.18-8.1.3.el5/kernel/drivers/s390/block/dasd_eckd_m
od.ko' [elf64-s390] to `/tmp/initrd.s14762/lib/dasd_eckd_mod.ko'
[elf64-s390]
copy from
`/lib/modules/2.6.18-8.1.3.el5/kernel/drivers/s390/block/dasd_fba_mo
d.ko' [elf64-s390] to `/tmp/initrd.s14762/lib/dasd_fba_mod.ko'
[elf64-s390]
Adding module jbd
Adding module ext3
Adding module dasd_mod with options dasd=0100,0150,0101,200
Adding module dasd_eckd_mod
Adding module dasd_fba_mod
```

Note: Please note the syntax of mkinitrd command is different between RHEL5 and SLES10.

Linux Logical Volume Manager

Linux systems support a nice feature to manage disks when you don't know at first how much space your partitions will consume: the Logical Volume Manager (LVM). LVM allows you to join disks together to form a single logical bigger disk, and then organize your partitions on this logical bigger disk. Figure 6-8 on page 309an example of managing disks under LVM.

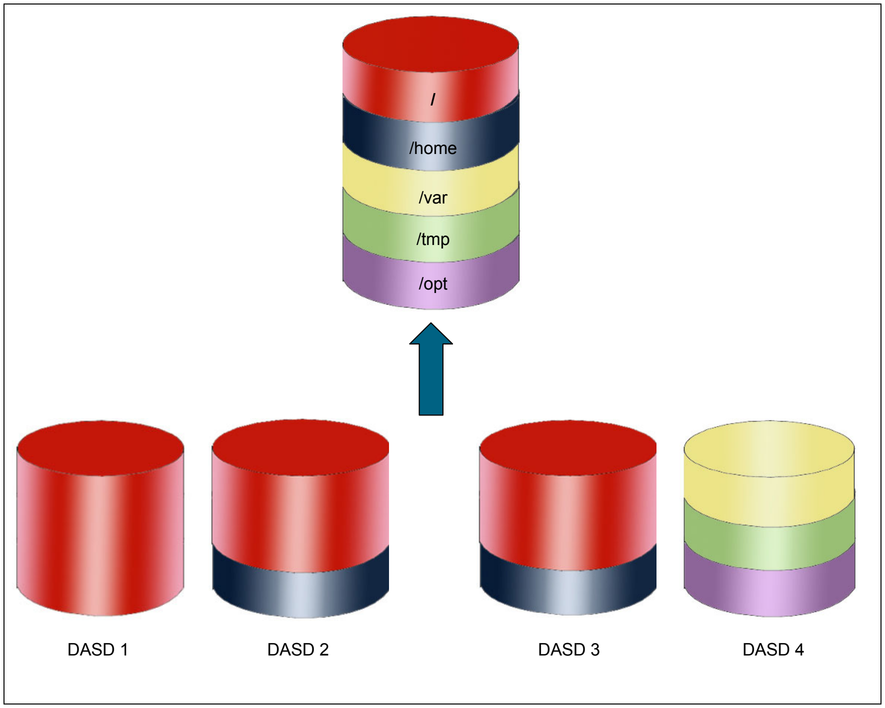

Figure 6-8 Linux LVM managing four DASDs.

As you can see on Figure 6-8, there are four DASDs (at the bottom) being used to form a single logical disk (at the top) that holds the system's partitions. The root filesystem (*/*) in that figure is *in reality* allocated on three different DASDs. The system doesn't really know or care about this since the LVM code handles address translations and accesses the data on the real disks. On Figure 6-8, you can see that the /home partition is also spread across two DASDs, and that the */var*, */tmp* and */opt* partitions are on a single DASD.

As you may recall from 4.4, "Installing Linux" on page 176, we chose to make room for its root filesystem over multiple 3390-3 volumes. Assume, for example, that the 3GB /home partition that we created just ran out of space and we wanted to give it some more space. The traditional way of getting along with this problem would be to bring a new DASD into the system, bigger than the one hosting the /home partition, and to move its contents to the new disk area. This is very awkward, time consuming and error prone. With LVM, though, this becomes an easy task.

Extending a *logical volume*, such as the /home partition of our Linux installation, is a matter of adding one more disk or stripe of a disk to the logical group so that this new space can be given to any of its logical volumes. In a rough analogy, this is similar to building a new room to the side of your living room and later breaking the wall apart to join them. You are left with the exact same living room and some extra space now. Figure 6-9 on page 310 explains this in terms of disks.

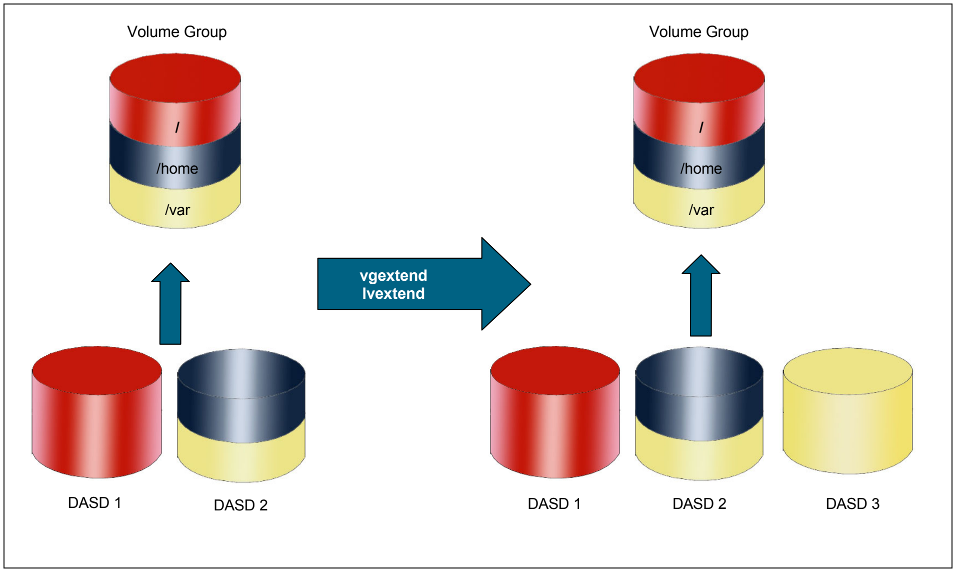

Figure 6-9 Expanding a Linux partition by adding more DASD to LVM

Some Linux distributions, such as YaST, provide mechanisms for users to friendly interact with LVM through a user interface. In systems where this is not available, there are commands to accomplish LVM management as well. The following are the most important ones you should know about.

- **pvcreate**
  - creates a *physical volume* entry on the disk, so that LVM knows about it.
- **vgscan**
  - scans all disks for volume groups and builds the files `/etc/lvmtab` and `/etc/lvmtab.d/*` which are the database for all other LVM commands
- **vgcreate**
  - creates a volume group using the disks passed as parameters. These disks must have been initialized with pvcreate.
- **vgextend**
  - adds a disks to a volume group. The disks must have been initialized with pvcreate.
- **lvcreate**

  – creates a logical volume inside a volume group.

- **lvremove**
  – removes a logical volume from a volume group. You should only use this on empty volumes or ones whose data is not of interest to you anymore.
- **lvextend**
  – extends a logical volume by using unallocated storage space in the volume group

To expand our 3GB /home partition, we would follow these steps:

1. As z/VM `MAINT`: attach another 3390-3 DASD to our z/VM system
2. As z/VM `MAINT`: define a minidisk on it and assign it to our Linux guest
3. As Linux `root`: use vgextend to add it to our volume group
4. As Linux `root`: use lvextend to expand our /home logical volume by using the newly added storage space to the volume group.

For a more detailed discussion on Linux LVM and a how-to for using it, refer to *Linux for IBM eServer zSeries and S/390: Distributions*, SG24-6264 or to http://tldp.org/HOWTO/LVM-HOWTO/. For a performance discussion on LVM itself, refer to http://www.ibm.com/developerworks/linux/linux390/perf/tuning_rec_dasd_optimizedisk.html and http://www.ibm.com/developerworks/linux/linux390/perf/tuning_rec_dasd_volMan.html

6.4.3 Network management

This section details the steps required to add an additional network interface to a running Linux Guest, this interface being connected to a VSWITCH, a GUEST-LAN or a Hipersocket network.

Once the network connectivity of the guest is established, Linux can be configured to make use of the connection. When running SLES10, the steps are detailed below.

1. Make sure the devices are seen by Linux, using for instance lscss command

Example 6-8 Checking the availability of the new devices

```
lnxguill:~ # lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.0191 0.0.0000  3390/0A 3990/E9      F0  F0  FF   52565A5E 00000000
0.0.0100 0.0.0001  3390/0C 3990/E9 yes  F0  F0  FF   5054585C 00000000
```

```
[...]
0.0.0200 0.0.0012  3390/0C 3990/E9      F0  F0  FF   5054585C 00000000
0.0.0D20 0.0.0013  1732/01 1731/01      80  80  80   21000000 00000000
0.0.0D21 0.0.0014  1732/01 1731/01      80  80  80   21000000 00000000
0.0.0D22 0.0.0015  1732/01 1731/01      80  80  80   21000000 00000000
```

2. Create a new configuration file in /etc/sysconfig/hardware

Example 6-9 Creating a new configuration file

```
lnxguill:/etc/sysconfig/hardware # cp -arv skel/hwcfg-qeth
hwcfg-qeth-bus-ccw-0.0.0d20
`skel/hwcfg-qeth' -> `hwcfg-qeth-bus-ccw-0.0.0d20'
```

3. Update the configuration file to reflect the Linux network hardware configuration, especially the CCW_CHAN_IDS and CCW_CHAN_MODE (used to specify a port name).

Example 6-10 Updating the network hardware configuration file

```
#!/bin/sh
#
# hwcfg-qeth
#
# Default configuration for a qeth device
# $Id: hwcfg-qeth 1069 2004-09-02 18:23:18Z zoz $
#

STARTMODE="auto"
MODULE="qeth_mod"
MODULE_OPTIONS=""
MODULE_UNLOAD="yes"

# Scripts to be called for the various events.
SCRIPTUP="hwup-ccw"
SCRIPTUP_ccw="hwup-ccw"
SCRIPTUP_ccwgroup="hwup-qeth"
SCRIPTDOWN="hwdown-ccw"

# CCW_CHAN_IDS sets the channel IDs for this device
# The first ID will be used as the group ID
CCW_CHAN_IDS="0.0.0d20 0.0.0d21 0.0.0d22"

# CCW_CHAN_NUM set the number of channels for this device
# Always 3 for an qeth device
CCW_CHAN_NUM=3
```

```
# CCW_CHAN_MODE sets the port name for an OSA-Express device
# CCW_CHAN_MODE="Z44LAN01"

# QETH_OPTIONS sets additional options for the OSA-Express or
# HiperSockets device. Valid parameters are:
# add_hhlen buffer_count broadcast_mode canonical_macaddr
checksumming
# fake_broadcast fake_ll priority_queueing route4 route6
# QETH_OPTIONS="fake_ll=1"

# QETH_IPA_TAKEOVER enables IP address takeover for this device
# QETH_IPA_TAKEOVER=0
```

Important: Devices numbers have to be lowercase.

4. Create the network configuration file in /etc/sysconfig/network:

Example 6-11 Creating the network configuration file

```
lnxguill:/etc/sysconfig/network # cp -arv
ifcfg-qeth-bus-ccw-0.0.0600 ifcfg-qeth-bus-ccw-0.0.0d20
`ifcfg-qeth-bus-ccw-0.0.0600' -> `ifcfg-qeth-bus-ccw-0.0.0d20'
```

5. Update the configuration file with your TCP/IP configuration

Example 6-12 Updating TCP/IP configuration

```
lnxguill:/etc/sysconfig/network # cat ifcfg-qeth-bus-ccw-0.0.0d20
BOOTPROTO="static"
UNIQUE=""
STARTMODE="onboot"
IPADDR="172.0.0.1"
NETMASK="255.255.255.0"
NETWORK="172.0.0.0"
BROADCAST="172.0.0.255"
```

6. Enable the network interface in Linux

Example 6-13 Enabling the network interface

```
lnxguill:/etc/sysconfig/hardware # echo '0.0.0d20,0.0.0d21,0.0.0d22'
> /sys/bus/ccwgroup/drivers/qeth/group
lnxguill:/etc/sysconfig/hardware # echo 1 >
/sys/devices/qeth/0.0.0d20/online
```

7. Make sure the devices have been brought online

Example 6-14 Checking the devices have been brought online

```
lnxguill:/etc/sysconfig/hardware # lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.0191 0.0.0000  3390/0A 3990/E9      F0  F0  FF   52565A5E 00000000
0.0.0100 0.0.0001  3390/0C 3990/E9 yes  F0  F0  FF   5054585C 00000000
[...]
0.0.0200 0.0.0012  3390/0C 3990/E9      F0  F0  FF   5054585C 00000000
0.0.0D20 0.0.0013  1732/01 1731/01 yes  80  80  80   21000000 00000000
0.0.0D21 0.0.0014  1732/01 1731/01 yes  80  80  80   21000000 00000000
0.0.0D22 0.0.0015  1732/01 1731/01 yes  80  80  80   21000000 00000000
```

8. Reload the network configuration, using the **rcnetwork service restart** command.

Example 6-15 Restarting the network configuration service

```
lnxguill:/etc/sysconfig/hardware # rcnetwork restart
Shutting down network interfaces:
    eth0
    eth0      configuration: qeth-bus-ccw-0.0.0600
done
    eth1
    eth1      configuration: qeth-bus-ccw-0.0.0d20
done
Shutting down service network  .  .  .  .  .  .  .  .  .  .  .  .  .
done
Hint: you may set mandatory devices in /etc/sysconfig/network/config
Setting up network interfaces:
    lo
    lo        IP address: 127.0.0.1/8
done
    eth0
    eth0      configuration: qeth-bus-ccw-0.0.0600
    eth0      IP address: 9.12.5.66/23
done
    eth1
    eth1      configuration: qeth-bus-ccw-0.0.0d20
    eth1      IP address: 172.0.0.1/24
done
Setting up service network  .  .  .  .  .  .  .  .  .  .  .  .  .  .
done
```

9. Check the Linux TCP/IP configuration, to make sure a new interface has been created.

Example 6-16 Checking the Linux TCP/IP configuration

```
lnxguill:/etc/sysconfig/hardware # ifconfig -a
[...]

eth1      Link encap:Ethernet  HWaddr 02:00:01:00:00:05
          inet addr:172.0.0.1  Bcast:172.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::200:100:100:5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:936 (936.0 b)

[...]
```

With RedHat 5, the steps are different, and detailed below:

1. Make sure the devices are seen by Linux, using for instance lscss command

Example 6-17 Checking the availability of the new devices

```
lnxguill:~ # lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.0191 0.0.0000  3390/0A 3990/E9      F0  F0  FF   52565A5E 00000000
0.0.0100 0.0.0001  3390/0C 3990/E9 yes  F0  F0  FF   5054585C 00000000
[...]
0.0.0200 0.0.0012  3390/0C 3990/E9      F0  F0  FF   5054585C 00000000
0.0.0D20 0.0.0013  1732/01 1731/01      80  80  80   21000000 00000000
0.0.0D21 0.0.0014  1732/01 1731/01      80  80  80   21000000 00000000
0.0.0D22 0.0.0015  1732/01 1731/01      80  80  80   21000000 00000000
```

2. Create a new alias for your network interface in file /etc/modprobe.conf

Example 6-18 Alias creation

```
[root@mbase40 etc]# cat modprobe.conf
options dasd_mod dasd=0100,0150,0101,200
alias eth0 qeth
alias eth1 qeth
```

This tells the kernel to use the qeth driver to configure the eth1 network interface.

3. Create the configuration file for the new network interface card, eth1:

Example 6-19 Creating the new the network configuration file

```
[root@mbase40 etc]# cat /etc/sysconfig/network-scripts/ifcfg-eth1
```

```
DEVICE=eth1
BOOTPROTO=static
BROADCAST=172.0.0.255
IPADDR=172.0.0.1
NETMASK=255.255.255.0
NETTYPE=qeth
NETWORK=172.0.0.0
ONBOOT=yes
PORTNAME=
SUBCHANNELS=0.0.0d20,0.0.0d21,0.0.0d22
TYPE=Ethernet
```

> **Important:** Devices numbers have to be lowercase.

4. Enable the network interface in Linux

Example 6-20 Enabling the network interface

```
[root@mbase40 etc]# echo '0.0.0d20,0.0.0d21,0.0.0d22' >
/sys/bus/ccwgroup/drivers/qeth/group
[root@mbase40 etc]# echo 1 > /sys/devices/qeth/0.0.0d20/online
```

5. Make sure the devices have been brought online

Example 6-21 Checking the devices have been brought online

```
[root@mbase40 etc]# lscss
Device   Subchan.  DevType CU Type Use  PIM PAM POM  CHPIDs
----------------------------------------------------------------------
0.0.0191 0.0.0000  3390/0A 3990/E9      F0  F0  FF   52565A5E 00000000
0.0.0100 0.0.0001  3390/0C 3990/E9 yes  F0  F0  FF   5054585C 00000000
[...]
0.0.0200 0.0.0012  3390/0C 3990/E9      F0  F0  FF   5054585C 00000000
0.0.0D20 0.0.0013  1732/01 1731/01 yes  80  80  80   21000000 00000000
0.0.0D21 0.0.0014  1732/01 1731/01 yes  80  80  80   21000000 00000000
0.0.0D22 0.0.0015  1732/01 1731/01 yes  80  80  80   21000000 00000000
```

6. Reload the network configuration, using the **rcnetwork service restart** command.

Example 6-22 Restarting the network configuration service

```
[root@mbase40 etc]# service network restart
Shutting down interface eth0:  [  OK  ]
Shutting down interface eth1:  [  OK  ]
Shutting down loopback interface:  [  OK  ]
Bringing up loopback interface:  [  OK  ]
```

```
Bringing up interface eth0:  [  OK  ]
Bringing up interface eth1:  [  OK  ]
```

7. Check the Linux TCP/IP configuration, to make sure a new interface has been created.

Example 6-23 Checking the Linux TCP/IP configuration

```
[root@mbase40 etc]# ifconfig -a
[...]

eth1      Link encap:Ethernet  HWaddr 02:00:01:00:00:05
          inet addr:172.0.0.1  Bcast:172.0.0.255  Mask:255.255.255.0
          inet6 addr: fe80::200:100:100:5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1492  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:12 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 b)  TX bytes:936 (936.0 b)

[...]
```

6.4.4 CPU management

When running Linux under z/VM, it is easy to dynamically - without need for reIPLing - add virtual CPUs to a guest that has been prepared accordingly.

Setup

LNXGUI DIRECTORY ENTRY

Example 6-24 Extract of LNXGUI DIRECTORY ENTRY

```
MACHINE ESA 4
CPU 00 BASE
CPU 01
```

LNXGUI directory a virtual machine running ESA architecture, supporting up to 4 CPUs. At IPL, two CPUs are defined in the guest.

LNXGUI initialization parameters

On the Linux side, it is necessary to tell the kernel that it can use more CPUs than it has effectively detected at IPL. This is done by adding the option additional_cpus into the zipl.conf parameter line, as illustrated below:

Example 6-25 LNXGUI zipl.conf

```
# Modified by YaST2. Last modification on Tue May 20 19:07:54 UTC 2008
[defaultboot]
defaultmenu = menu

[...]

###Don't change this comment - YaST2 identifier: Original name: ipl###
[ipl]
    target = /boot/zipl
    image = /boot/image
    ramdisk = /boot/initrd,0x1000000
    parameters = "root=/dev/dasda1   TERM=dumb additional_cpus=2"
[...]
```

After the file /etc/zipl.conf has been updated, it is mandatory to update the boot record to reflect the new configuration, running the command zipl. Reboot Linux for changes to take effect.

Adding a virtual CPU to a running Linux guest

If the guest is correctly setup as explained earlier, it is possible to add CPUs to a running Linux guest.

First step is to define a new CPU in the Linux virtual machine, as detailed in Example 6-26, using the class G command DEFINE CPU:

Example 6-26 Define a new virtual CPU in the Linux virtual machine

```
00: CPU 00  ID  FF04DE5020978000 (BASE) CP   CPUAFF ON
00: CPU 01  ID  FF04DE5020978000 CP   CPUAFF ON
00:
00: CP DEFINE CPU 02
00: CPU 02 defined
00:
00: CP Q CPUS
00: CPU 00  ID  FF04DE5020978000 (BASE) CP   CPUAFF ON
00: CPU 01  ID  FF04DE5020978000 CP   CPUAFF ON
00: CPU 02  ID  FF04DE5020978000 CP   CPUAFF ON
```

The virtual machine had two CPUs active at IPL, it now has three. The Linux guest has no knowledge yet of this extra CPU power available, as shown below:

Example 6-27 Initial CPU configuration

```
lnxguill:~ # cat /proc/cpuinfo
```

```
vendor_id       : IBM/S390
# processors    : 2
bogomips per cpu: 3735.55
processor 0: version = FF,  identification = 04DE50,  machine = 2097
processor 1: version = FF,  identification = 04DE50,  machine = 2097
```

The remaining step is to activate the CPU in Linux. To do so, issue a command similar to the command shown in Example 6-28 on page 319:

Example 6-28 Enabling the new CPU in Linux

```
lnxguill:~ # echo 1 > /sys/devices/system/cpu/cpu2/online
```

The message in Example 6-29 on page 319 will show up in the log files when the CPU has been activated:

Example 6-29 CPU activation log message

```
Jun 12 15:04:00 lnxguill kernel: cpu 2 phys_idx=2 vers=FF ident=04DE50
machine=2097 unused=8000
```

To check the number of CPUs seen by Linux, the administrator can use the cat /proc/cpuinfo command, as shown in

Example 6-30 Querying the number of CPUs

```
lnxguill:~ # cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 3
bogomips per cpu: 3735.55
processor 0: version = FF,  identification = 04DE50,  machine = 2097
processor 1: version = FF,  identification = 04DE50,  machine = 2097
processor 2: version = FF,  identification = 04DE50,  machine = 2097
```

7

Performance Monitoring and System Analysis

In this chapter, we introduce z/VM scheduling concepts, and monitoring commands, and relate those commands to z/OS equivalents. We also introduce some z/VM monitoring tools, as well as performance tuning concepts.

The chapter also covers concepts and techniques related to monitoring Linux running on System z.

Objective:

After completing this chapter, you will be able to:

- Understand the basics of z/VM scheduling concepts.
- Understand the usage of CP commands related to performance Monitoring.
- Relate to z/OS commands and monitoring concepts.
- Understand what monitoring tools are available for z/VM
- Develop an understanding on system monitoring best practices.

7.1 Monitoring z/VM

In this section, you will be introduced to z/VM scheduling basics. You will also learn about System z/VM and Linux monitoring commands.

There are many aspects to be considered when looking at the performance of a z/VM system, including the following areas:

- User response time
- Throughput
- Device utilization
- Number of users supported
- Reliability
- System capacity

If you are familiar with z/OS, you may have heard about or worked with the z/OS Work Load Manager (WLM). The z/OS Workload Manager (WLM) component provides the ability to dynamically allocate or re-distribute server resources, such as CPU, I/O, and memory across a set of workloads based on user defined goals and their resource demand within a z/OS image.

For more information on the z/OS Work Load Manager, refer to http://www-03.ibm.com/servers/eserver/zseries/zos/wlm/pdf/zWLM.pdf

z/VM Scheduling

To understand z/VM performance parameters and factors, it is important to be aware of the underlying system scheduling.

In z/OS, Job Control Language (JCL) is used to tell the system which programs to execute, followed by a description of program inputs and outputs. JCL consists of a series of statements, each of which provides specific instructions or information for batch processing jobs. It provides the system with a source of information about details such as the programs to execute, the location of required data sets, the department to be billed for CPU processing time, and the job priority. The system uses a job entry subsystem (JES) to receive jobs into the operating system, schedule them for processing by MVS, and control their output processing. JES2 and the base control program (BCP) provide the necessary functions to get jobs into, and output out of the MVS system.

z/VM employs similar scheduling concepts where the Control Program and the Scheduler cooperate to perform the scheduling tasks which involves a number of components illustrated in Figure 7-1 on page 323.

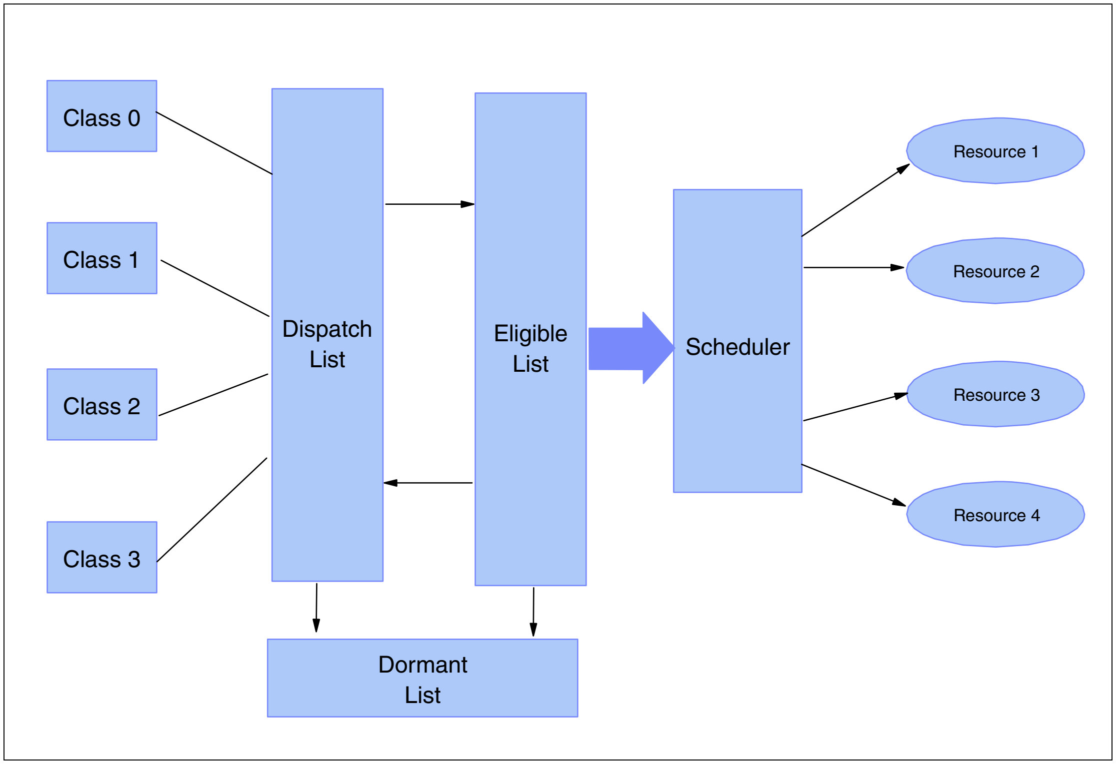

Figure 7-1 z/VM scheduling components

A definition of some of the scheduling components and how it all works are described here.

Dormant List: If any virtual machine is not doing any processing, the Control Program moves it to the Dormant List. The virtual machine is now “Asleep”.

Eligible List: A sleeping virtual machine can “Wake Up” in response to an event, such as pressing the enter key on the console for example. The virtual machine is then moved to the Eligible List and it becomes *eligible* to consume system resources.

The Scheduler: The scheduler examines the list of eligible virtual machines before they are given system resources. Specifically, the scheduler looks into the virtual machines definition information (VMDBK) for resources that have been consumed in the past, like processor, storage, and paging.

The Dispatch List: If the scheduler determines that there are enough available resources to satisfy your needs without putting any other virtual machine in jeopardy, the virtual machine will be moved to the dispatch list.

Time Slice: Virtual machines in the dispatch list wait in a queue for their share of CPU time, usually referred to as a time slice. Virtual machines who have more work to do can stay in the dispatch list to get another time slice.

User Classes: When you first log on to z/VM, the Control Program creates an area of storage that contains information pertinent to the operation that you are associated with. This storage area is called a Virtual Machine Descriptor Block (VMDBK). Pointers to this block get moved around the lists illustrated in Figure 7-1 on page 323.

z/VM categorizes users under four distinct classes. User class affects how the scheduler handles the user's scheduling scheme. For example, users in class 1 are assumed to be running short processing transactions. Users in this class are given times slices and are expected to complete their processing within that time slice. However, if users have visited the dispatch list few times and have not completed their processing, the dispatcher can upgrade the user to class 2. Class 2 users are allowed to stay in dispatch list for longer intervals to finish their medium-length transactions.

Table 7-1 on page 324 illustrates the different user classes implemented in z/VM along with their description.

Table 7-1 User Classes and their description

User class	Description
Class 0	This class indicates the users that were added to the dispatch list with no delay in the eligible list, regardless of the length of their current transaction.
Class 1	This class indicates the users that have just begun a transaction, and therefore are assumed to be currently processing short transactions.
Class 2	This class indicates the users that did not complete their current transactions during their first dispatch list stay and therefore are assumed to be running medium-length transactions.
Class 3	This class indicates the users that did not complete their current transactions during their second dispatch stay and therefore are assumed to be running long transactions.

If you have command privilege class E, you can issue the **`CP INDICATE LOAD`** command to view information about these classes of user; see Figure 7-2 on page 325.

Qn indicates users in the dispatch list. **E**n indicates users in the eligibility lists where n is the class of the user 0, 1, 2, or 3.

```
ind
AVGPROC-000% 04
XSTORE-000000/SEC MIGRATE-0000/SEC
MDC READS-000001/SEC WRITES-000000/SEC HIT RATIO-084%
PAGING-0/SEC STEAL-000%
Q0-00000(00000)                           DORMANT-00016
Q1-00000(00000)           E1-00000(00000)
Q2-00000(00000) EXPAN-001 E2-00000(00000)
Q3-00000(00000) EXPAN-001 E3-00000(00000)

PROC 0000-000% CP       PROC 0001-000% CP
PROC 0002-000% CP       PROC 0003-000% CP

LIMITED-00000
```

Figure 7-2 INDicate LOAD command

> **Note:** Any values in the **E**n fields normally indicates a performance problem (Probably a storage or paging constraint).

7.1.1 CP Monitoring Commands

This section introduces some basic z/VM commands. If you are familiar with z/OS, you will find references to the equivalent or similar commands of z/OS to help you expland on your existing knowledge of z/OS. If you are not familiar with z/OS, you will learn the basic functionality available to you in z/VM.

CP Commands

Performance management in a z/VM system requires tools to collect and analyze data, and to control resource usage by virtual machines. These tools are provided by the Control Program (CP) by means of commands and monitoring functions.

There are four CP command groups which will be discussed:

- CP INDICATE
- CP QUERY
- CP MONITOR
- CP SET

We present these commands and their most commonly used parameters.

CP INDICATE

The CP INDICATE command gives a snapshot of resource utilization. Because it is only a snapshot, however, commands need to be issued several times and the results examined for changes.

Some of the commands have an EXPanded option, which will give additional detail. The INDICATE command can be used by system resource operators, system programmers, system analysts, and general users (class B, C, E, and G users) although different classes of user will see different results to some commands.

z/OS analgoy: CP INDICATE command is similar to DISPLAY ACTIVE (or its abbreviation da).

CP INDICATE ACTIVE

Use this command to do the following:

- Display the total number of users active in a specified time interval.
- Display the number of users in the dispatch, eligible, and dormant lists that were active in a specified time interval.

Operands

nnnnn identifies an integer number specifying the time interval. The valid time period is 0 to 9 hours or 540 minutes or 32400 seconds. The default is 60 seconds.

SEC identifies that *nnnnn* was specified in seconds. This is the default.

MIN identifies that *nnnnn* was specified in minutes.

HRS identifies that *nnnnn* was specified in hours.

A sample of this command can be seen in Example 7-1 on page 327.

Example 7-1 INDicate ACTive command

```
ind active
0009 USERS, 0005 DISP, 0000 ELIG, 0004 DORM
Ready; T=0.01/0.01 09:54:21
```

CP INDICATE I/O

Use this command to identify the virtual machines currently in an I/O wait state and the real I/O device number that they are waiting on. This command can assist you in spotting a volume for which minidisk cache or CU cache were mistakenly turned off.

CP INDICATE LOAD

Use this command to display information about system resources. For a general user, CP displays a subset of that information. Note that the processor usage information shown by this command is actually a “smoothed” average over a period of time. Changes in system load will likely take few minutes to influence this number. This command is a good starting point to analyze system performance problems.

Sample output from this command can be seen in Example 7-2 on page 327.

Example 7-2 CP INDicate LOAD

```
cp ind load
AVGPROC-000% 04
XSTORE-000000/SEC MIGRATE-0000/SEC
MDC READS-000001/SEC WRITES-000000/SEC HIT RATIO-100%
PAGING-0/SEC STEAL-000%
Q0-00001(00000)                           DORMANT-00010
Q1-00001(00000)           E1-00000(00000)
Q2-00000(00000) EXPAN-001 E2-00000(00000)
Q3-00000(00000) EXPAN-001 E3-00000(00000)

PROC 0000-000% CP       PROC 0001-000% CP
PROC 0002-000% CP       PROC 0003-000% CP

LIMITED-00000
Ready; T=0.01/0.01 09:56:25

                                                          RUNNING
VMLINUX6
```

Note: For monitoring instantaneous processor load, you should use the Performance Tool Kit.

Similarly, on z/OS you would use the command DISPLAY ACTIVE to display such information.

CP INDICATE PAGING

Use this command to display:

- A list of the virtual machines in page wait status
- Page residency data for all system users

z/OS analogy: CP INDICATE PAGING corresponds to z/OS command D ASM

CP INDICATE QUEUES

Use this command to display, in order of their priority, current members of the dispatch and eligible lists. If users have a virtual multiprocessor, you may see more than one entry for a single user.

Example 7-3 INDicate Queues command

```
ind queues
MAINT          Q1 R00 00001400/00001357 LNXGUI   MP01 Q3 PS  00000000/00000000
LNXGUI         Q3 PS  00020637/00020623 LNXKEN   MP01 Q3 PS  00000000/00000000
LNXKEN         Q3 PS  00114857/00114843 LNXCER        Q3 PS  00108507/00108493
LNXCER   MP01 Q3 PS  00000000/00000000 TCPIP         Q0 PS  00003011/00002656
Ready; T=0.01/0.01 10:10:01
```

This command corresponds to z/OS DISPLAY ACTIVE command.

CP QUERY SPACES

Use this command to display information about a user's address space and paging usage.

Example 7-4 INDicate SPaces command

```
ind spaces

Spaceid=MAINT:BASE  Owned size=123M  PRIVATE
Pages:  Main=1400  Xstore=0  Dasd=0  Locked=---
Private paging:
  Xstore:  Reads=0          Writes=0          Migrates=0
  Dasd:    Reads=1          Writes=1
Shared paging:
  Xstore:  Reads=0          Writes=0          Migrates=0
  Dasd:    Reads=0          Writes=0
Ready; T=0.01/0.01 10:12:58
```

This commands corresponds to z/OS DISPLAY ACTIVE command.

CP QUERY

The Query commands can be used to display a great deal of information about the system and users. We introduce some query commands that may be useful when investigating performance problems. If you have z/OS background, you will benefit from z/OS hints related to z/VM commands.

CP QUERY ALLOC

Use this command to display the number of cylinders or pages that are allocated, in use, and available for DASD volumes attached to the system. Use Q ALLOC PAGE for detailed information about paging space

Note: Do not confuse this command with z/OS ALLOCATE command. These two commands are totally different, despite the name similarity.

z/OS analogy: CP QUERY ALLOC command corresponds to z/OS D ASM command

Example 7-5 Query ALLOC command

```
q alloc
DASD 1A20 LX6RES 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     PAGE  TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     SPOOL TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     DRCT  TOTAL=00000000020 INUSE=00000000001 AVAIL=00000000019, ACTIVE
DASD 1A21 LX6SPL 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     PAGE  TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     SPOOL TOTAL=00000003338 INUSE=00000000803 AVAIL=00000002535
     DRCT  TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
DASD 1A22 LX6PAG 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     PAGE  TOTAL=00000003338 INUSE=00000000000 AVAIL=00000003338
     SPOOL TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     DRCT  TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
DASD 1A23 LX6W01 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     PAGE  TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     SPOOL TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     DRCT  TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
DASD 1A24 LX6W02 3390 CKD-ECKD (UNITS IN CYLINDERS)
     TDISK TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     PAGE  TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     SPOOL TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
     DRCT  TOTAL=00000000000 INUSE=00000000000 AVAIL=00000000000
......
IPL NUCLEUS ACTIVE ON VOLUME LX6RES
Ready; T=0.01/0.01 10:16:51
```

CP QUERY CHPIDS

Use this command to display all of the 256 machine's channel paths and their physical status. I/O performance problems can occur if CHPIDs are offline or not available

Example 7-6 Query CHPIDS command

```
q chpids
    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0x  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
1x  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
2x  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
3x  -  +  .  .  .  .  .  .  .  .  +  +  +  +  .  .
4x  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
5x  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
6x  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +
7x  +  +  +  +  +  +  +  +  +  +  +  +  .  .  .  .
8x  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
9x  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Ax  .  .  .  +  .  .  .  +  .  .  .  .  .  .  .  .
Bx  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Cx  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Dx  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Ex  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
Fx  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +  +

+  Available
-  Offline
.  Not configured
Ready; T=0.01/0.01 10:18:45
```

z/OS analogy: This command acts like the z/OS */D M=CHP* command.

Both commands display machine's channel paths and their physical statuses.

CP QUERY CPLOAD

Use this command to display information regarding the last CP IPL. The information displayed includes the location of the CP module that was last used, the location of the parm disk, and how CP was started.

Example 7-7 Query CPLOAD command

```
q cpload
Module CPLOAD was loaded from minidisk on volume LX6RES at cylinder 39.
Parm disk number 1 is on volume LX6RES, cylinders 39 through 158.
Last start was a system IPL.
Ready; T=0.01/0.01 10:21:13
```

z/OS analogy: CP QUERY CPLOAD is very similar to /D IPLINFO command on z/OS.

CP QUERY CPOWNED

Use QUERY CPOWNED to display the list of CP-owned DASD volumes. If paging or spooling problems occur, this can be checked to ensure that all required volumes are available. Figure 7-8 on page 332 illustrates sample output of cp query cpowned command.

Example 7-8 cppwned sample output

```
q cpowned
Slot  Vol-ID  Rdev  Type   Status
   1  LX6RES  1A20  Own    Online and attached
   2  LX6SPL  1A21  Own    Online and attached
   3  LX6PAG  1A22  Own    Online and attached
   4  LX6W01  1A23  Own    Online and attached
   5  LX6W02  1A24  Own    Online and attached
   6  DK8226  8226  Own    Online and attached
   7  DK8227  8227  Own    Online and attached
   8  DK8228  8228  Own    Online and attached
   9  ------  ----  -----  Reserved
  10  ------  ----  -----  Reserved
```

CP QUERY FRAMES

Use QUERY FRAMES to display the status of host real storage. It may be necessary to do this if you are getting users in the eligible list when you need to check storage utilization.

Example 7-9 Query FRAMES command

```
cp q frames
All Frames:
   Configured=1048575  Real=1048575  Usable=1048575  Offline=0
   Pageable=1026485  NotInitialized=0  GlobalClearedAvail=128
   LocalClearedAvail=128  LocalUnclearedAvail=124

Frames < 2G:
   GlobalUnclearedAvail=454261  Pageable=521003  LogicalFreeStorage=56
   RealFreeStorage=4  LockedRS=348  LockedCmd=0
   MinidiskCache=9339  Nucleus/Prefix=2430  Trace=325  Other=122

Frames > 2G:
   GlobalUnclearedAvail=436453  Pageable=505482  LogicalFreeStorage=1194
   RealFreeStorage=19  LockedRS=57  LockedCmd=0
   MinidiskCache=13138  Other=17536
```

z/OS analogy: CP QUERY FRAMES is similar to *z/OS /D - M=STOR* command

CP QUERY MAXUSERS

Use QUERY MAXUSERS to display the maximum number of logged-on users allowed. If users cannot logon, this may be a good command to execute.

z/OS analogy: To display the number of maximum users in z/OS environment, you need to browse your system parameters file (sys/. parmlib IEASYSxx)

CP QUERY MDC

Use this command from a Class B user to query minidisk cache (MDC) settings for the entire system, for a real device, an active minidisk, or a minidisk defined in the directory. This command is useful for investigating I/O problems

Example 7-10 Query MDCache command

```
q mdc
Minidisk cache ON for system
Storage MDC min=0M max=128M, usage=3%, bias=1.00
Xstore MDC min=0M max=0M, usage=0%, bias=1.00
Ready; T=0.01/0.01 10:23:24
```

CP QUERY NAMES

Use QUERY NAMES to display:

- A list of all logged-on users.
- The real or logical device number of the display that each user is connected.

Example 7-11 Query Names command

```
q n
LNXCER   - DSC , LNXGUI   - DSC , LNXKEN   - DSC , COSTA    -L0004
FTPSERVE - DSC , TCPIP    - DSC , DTCVSW2  - DSC , DTCVSW1  - DSC
VMSERVR  - DSC , VMSERVU  - DSC , VMSERVS  - DSC , OPERSYMP - DSC
DISKACNT - DSC , EREP     - DSC , OPERATOR - DSC , MAINT    -L0003
VSM     - TCPIP
Ready; T=0.01/0.01 10:26:04
```

z/OS analogy: On **z/OS systems**, you can display similar information by running the *Display Active* command.

CP QUERY QIOSSIST

Use QUERY QIOASSIST to determine the current status of the queue-I/O assist for a virtual machine.

Example 7-12 Query QIOASSIST command

```
q qioassist
ALL USERS SET - ON

USER      SETTING   STATUS
MAINT     ON        INACTIVE
Ready; T=0.01/0.01 10:28:40
```

Use this command when investigating I/O or networking problems.

CP QUERY QUICKDSP

Use QUERY QUICKDSP to display the QUICKDSP attribute for a user.

Figure 7-13 on page 334 illustrates a sample output of the QUICKDSP command.

Example 7-13 QUICKDSP sample output

```
cp query quickdsp maint
USER MAINT   :  QUICKDSP = OFF
```

CP QUERY RESERVED

Use this command to display the number of reserved real storage frames. You can use SET RESERVED to reserve pages of storage for a user.

Example 7-14 Query REServed command

```
q res
MAINT    RSV=00000010 ACT=00000010
LNXGUI   RSV=00000020 ACT=00000020
REQUESTED FRAME TOTAL=00000030; ACTUAL FRAME TOTAL=00000030
Ready; T=0.01/0.01 10:57:15
```

These commands can be used when tuning users in a storage-constrained system.

CP QUERY SRM

Use QUERY SRM (System Resource Manager) to display system-wide parameters used by the scheduler to set the priority of system resource access.

The CP SET SRM command is useful if you need to control a user's use of resources, depending on the class of user.

Example 7-15 Query SRM command

```
q srm
IABIAS : INTENSITY=90%; DURATION=2
LDUBUF : Q1=100% Q2=75% Q3=60%
STORBUF: Q1=300% Q2=250% Q3=200%
DSPBUF : Q1=32767 Q2=32767 Q3=32767
DISPATCHING MINOR TIMESLICE = 5 MS
MAXWSS : LIMIT=9999%
...... : PAGES=999999
XSTORE : 0%
Ready; T=0.01/0.01 12:27:28
```

Use this command carefully and monitor its usage because eligible lists can occur if not used properly.

z/OS analogy: In z/OS, you can set SRM parameters, but you can not view these parameters in a direct way.

CP QUERY STOR

Use QUERY STORage to display the size of real storage.

Example 7-16 Query STORage command

```
q stor
STORAGE = 4G
Ready; T=0.01/0.01 12:35:16
```

z/OS analogy: In z/OS, you can run the command **-D M=STOR** to display the size of the real storage.

CP QUERY SYSTEM

Use QUERY SYSTEM to display current user access to a system DASD volume.

z/OS analogy: In z/OS, you have to know the device Number beforehand. You then run the z/OS command **-D U,, ALLOC, *devise number, start device number.***

CP QUERY XSTOR

Use QUERY XSTORAGE or QUERY XSTORE to display the assignment of real Expanded Storage.

Example 7-17 Query XSTORage

```
q xstor
XSTORE= 2048M online= 2048M
XSTORE= 2048M userid= SYSTEM usage= 0% retained= 0M pending= 0M
XSTORE MDC min=0M, max=0M, usage=0%
XSTORE= 2048M userid=  (none)  max. attach= 2048M
Ready; T=0.01/0.01 12:35:19
```

Use this command to investigate response time or paging problems.

CP Monitor

The CP Monitor facility collects system performance data that can be made available to an external data reduction program for analysis. Statistics related to system operation or contention for major system resources can be generated. These resources include processors, storage, I/O devices, and the paging subsystem. You can control the amount and nature of the data collected.

In general, monitoring is performed in this order:

1. The user employs the privileged CP MONITOR command to control monitoring. This includes the type, amount, and nature of data to be collected.

2. The monitor collects performance data and stores monitor records in a saved segment.

3. A CMS application program connects to the CP MONITOR System Service to establish a data link with CP.

4. The application retrieves and processes monitor records from the saved segment.

CP SET commands

CP SET commands can change the performance characteristics of the entire system or of a single user. The next subsection introduces these commands.

CP SET MAXUSERS

Use this command to control the number of allowed users to log on. If the CPU is constrained, you could use this command to limit the number of users.

> **z/OS analogy:** In z/OS, to set the number of users, you would set the parameters in the member IEASYSxx in the ParmLib.

CP SET MDC

Use this command from a class B user to:

- Change minidisk cache settings for the entire system, for a real device, or for an active minidisk.
- Purge the cache of data from a real device or an active minidisk.
- Change a user's ability to insert data into the cache.

This command can be useful if you do not have much minidisk activity and you want to release some storage.

CP SET QIOASSIST

Use this command to control the queue-I/O assist (QDIO performance assist for V=V guests) for a virtual machine. This interpretive-execution assist applies to devices that use the Queued Direct I/O (QDIO) architecture, HiperSockets devices, and FCP devices.

CP QUICKDSP

Use this command to assign or unassign a user's immediate access to system resources.

> **z/OS analogy:** In z/OS, you would use WLM to perform similar tasks.

CP SET RESERVED

Use this command to establish the number of real storage frames that are available to a specific virtual machine.

This command may be useful when you are trying to tune specific guests in a storage-constrained environment.

CP SET SHARE

Use this command to change the system-resource-access priority for users. This is a complex command that can have profound effect on system and user resource usage. Use with care.

CP SET SRM

Use this command to change system parameters. These parameters define the size of the time slice and the access to resources for different user classes as seen by the scheduler. This is a complex command. The command's effects should be clearly understood before use.

> **Note:** For more information on SET SRM, refer to the z/VM: CP Commands and Utilities Reference which can be found at:
>
> http://publib.boulder.ibm.com/infocenter/zvm/v5r3/topic/com.ibm.zvm.v53.hcpb7/hcse4b21.htm

CP SET THROTTLE

Use this command to control the number of I/O operations that a guest operating system can initiate to a specific real device. This prevents a guest from interfering with, or dominating, I/O resources.

7.2 Tuning Basics

System z processors have evolved over the years to be highly robust and reliable. We present in this section some general tuning concepts and explore their practical implications on z/VM.

7.2.1 System tuning approach

Performance analysis and tuning is a multi-step process. Regardless of which tools you choose, the best methodology for analyzing the performance of a system is to start from the outside and work your way down to the small tuning details. Start by gathering data about the overall health of systems hardware and processes. How busy is the processor during the peak periods of each day? What happens to I/O response times during those peaks? Do they remain fairly consistent, or do they elongate? Does the system get memory constrained every day, causing page waits? Can current system resources provide user response times that meet service level agreements? Following a good performance analysis process can help you answer those questions.

It is important to know what tuning tools are available and the types of information that they provide. Equally important is knowing when to use those tools and what to look for. Waiting until the telephone rings with user complaints is too late to start running tools and collecting data. How will you know what is normal for your environment and what is problematic unless you check the system activity and resource utilization regularly? Conducting regular health checks on a system also gives you utilization and performance information that can be used for capacity planning purposes.

7.2.2 Where tuning can help

Even when the different workloads add up to less than the total amount of resources available, you might find that you are still unable to run the workload with good performance. The reason for that might be that the system is short on one specific resource. In such a situation, proper tuning can make a difference. Before tuning the system and workload, you need to understand what resource is the limiting factor in your configuration. Tuning changes tend to fall into one of these categories:

- Use less constrained resources.

One of the benefits of running Linux systems under z/VM is the ability to share and to overcommit hardware resources. The amount of memory that Linux thinks it owns is called virtual memory. The sum of the amounts of virtual memory allocated to each Linux guest can be many times the amount of real memory available on the processor. z/VM efficiently manages memory for Linux guests. With a system that is really memory constrained, one option might be to reduce overall z/VM memory usage by reducing the virtual machine size of Linux guests.

- Get a larger share of a constrained resource.

It is easy for users to increase the virtual memory size for Linux guests, since it is a quick change to the guest definition under z/VM. However, needlessly increasing virtual memory allocations can cause excessive paging (also known as thrashing) for the entire system. In the case of a system that is truly memory constrained, and where Linux virtual memory sizes have been assigned judiciously, consider reserving some memory pages for one particular Linux virtual machine, at the expense of all others. This can be done with a z/VM command (cp set reserved).

- Increase total available resources.

The most obvious approach to solving memory issues is to buy more hardware. The cost of memory has declined over the years, so that may be a good option. However, additional resources can be made available by stopping unneeded utility services in the Linux systems. Be aware that tuning does not increase the total amount of system resources. It simply allows those resources to be used more effectively for critical workloads.

7.2.3 Where tuning does not help

It is important to understand that you cannot run more work than you can fit in the machine. If your System z machine has two CPUs and the workload you want to run consists of three Linux virtual machines running WebSphere, where each of them runs a CPU for 100% all day, then it will not fit. There is no z/VM tuning that will make it fit, but there might be performance issues in the applications themselves to make the workload use less than 100% all day.

When a System z machine has four CPUs and the workload consists of three Linux virtual machines that each use a CPU for 100% all day, there is little to tune. In this case, z/VM has sufficient resources to give each Linux virtual machine what it requires (though one might want to look into changes to the configuration that allow you to use all four CPUs and make things run faster).

7.3 Performance and Monitoring tools for z/VM

The use of tools to monitor and enhance performance is common. Performance monitoring tools provides more functionality and reports on your system z/VM status.

In this section, we introduce some of these performance monitoring tools. For more details, please also see the IBM Redbooks publication *Linux on IBM System z: Performance Measurement and Tuning*, SG24-6926 .

7.3.1 IBM Performance Toolkit for VM

The IBM Performance Toolkit for VM is an enhanced real-time performance monitor and full screen operator that allows system programmers to monitor system performance and to analyze bottlenecks. The toolkit can help system programmers make more efficient use of system resources, increase system productivity, and improve user satisfaction. In addition to analyzing z/VM performance data, the Performance Toolkit for VM can process Linux performance data obtained from the IBM Resource Management Facility (RMF) and the Linux performance gatherer, rmfpms. Some of the other functions provided by the Performance Toolkit for z/VM include:

- Operation of the system operator console in full-screen mode
- Management of multiple z/VM systems (local or remote)
- Post-processing of performance toolkit for VM monitor data captured by the MONWRITE utility
- Viewing of performance monitor data using either Web browsers or PC-based
- 3270 emulator graphics
- TCP/IP performance reporting

Figure 7-3 on page 341 depicts a typical Performance Toolkit screen.

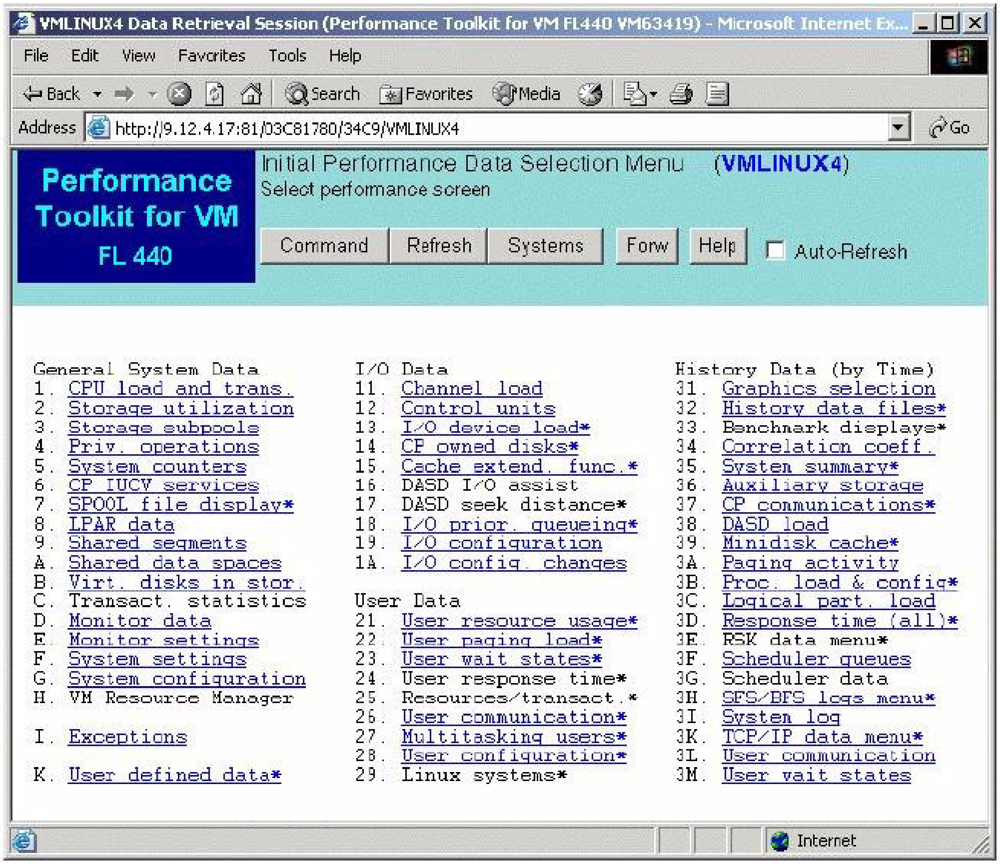

Figure 7-3 Performance toolkit

> **Note:** For more information on Performance Toolkit, refer to *Linux on IBM eServer zSeries and S/390: Performance Toolkit for VM*, SG24-6059

7.3.2 Tivoli Omegamon for z/VM and Linux

OMEGAMON® is a real-time software performance monitor for the VM (Virtual Machine) operating system. It runs under the Conversational Monitor System (CMS) operating system. OMEGAMON warns you of exceptional conditions automatically, and also displays the status of VM internal operations and resources in real time.

All of the OMEGAMON features and facilities are designed around the concept of a logical tuning approach for improving the performance of your system. The logical tuning approach consists of these steps:

- Defining standards for VM performance at your installation.
- Monitoring your system to measure actual performance against these
- standards.
- Identifying the cause of performance problems.
- Initiating action to correct performance problems.

Figure 7-4 on page 342 illustrates the overview of Tivoli® Omegamon guest performance view.

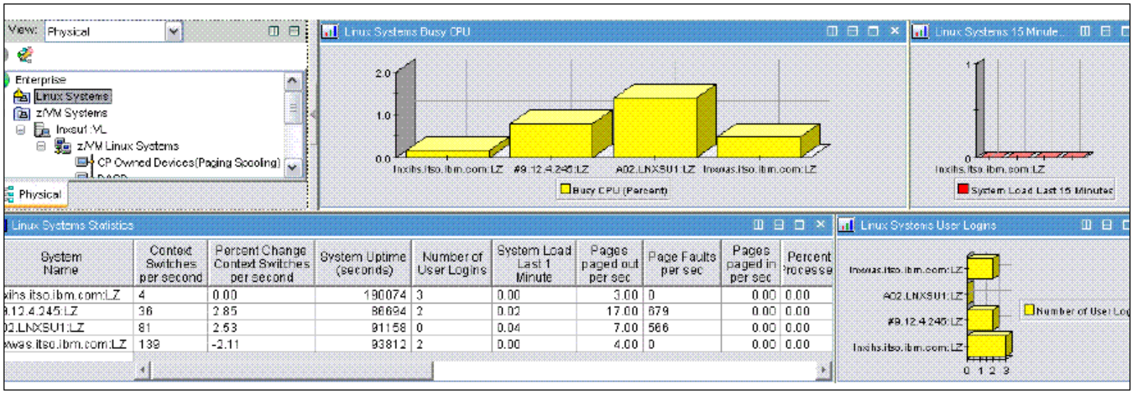

Figure 7-4 Tivoli Omegamon guest performance view

> **Note:** For additional information on IBM Tivoli Omegamon, refer to:
>
> - Linux on IBM System z: Performance Measurement and Tuning.
> - http://www-306.ibm.com/software/sysmgmt/products/support/IBMTivoliOMEGAMONonzVMLinux.html

7.3.3 IBM Tivoli Performance Modeler

IBM Tivoli Performance Modeler for z/OS V2.3 is a PC-based performance modeling and capacity planning tool that runs on MicroSoft Windows. It can be as portable as your PC or mobile computer. IBM Tivoli Performance Modeler for z/OS V2.3 is designed for system management on the IBM eServer zSeries and S/390.

With growing operating system complexity and the huge impact of responding to workload changes, basic Central Processor Unit (CPU) utilization is generally no longer sufficient information for performing capacity planning. IBM Tivoli Performance Modeler for z/OS V2.3 can be used to model the impact of changing:

- Number and speed of CPUs
- Disk I/O response times
- Paging rates (auxiliary and expanded memory paging)
- Logical partition (LPAR) definitions and parameter changes

The Tivoli Performance Modeler takes input from z/OS images, including system and application performance data, as a basis for the modeling program. The user can then make changes to the type of processor and modify the workload configuration to create a new model. The modeler then creates a series of graphs to depict the modeled workload.

Figure 7-5 on page 344 shows a complex workload of batch, database, and online applications.

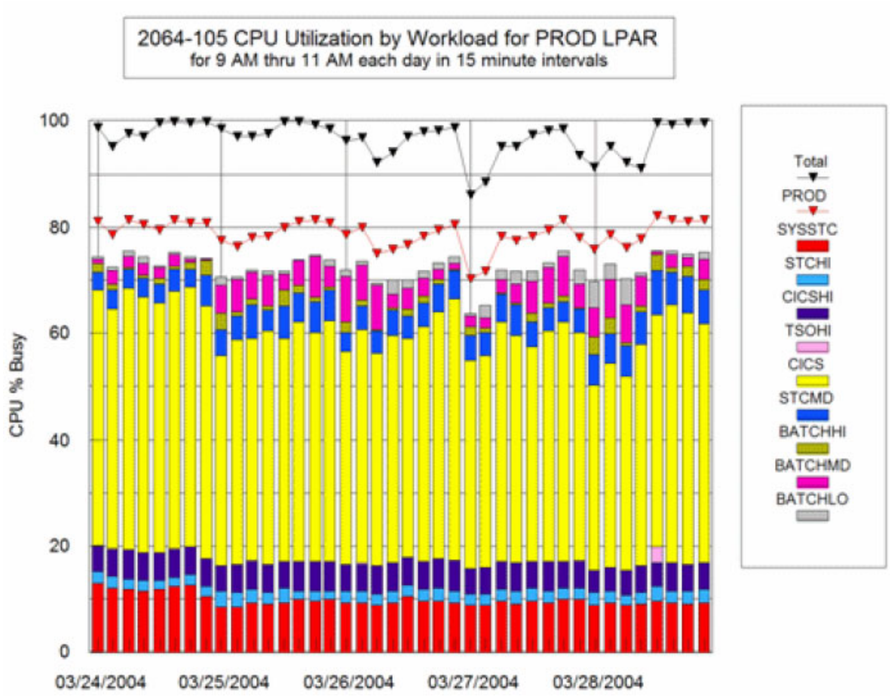

Figure 7-5 Tivoli Performance Modeler

7.3.4 IBM z/VM Planner for Linux Guests on IBM System z Processors

The z/VM Planner for Linux Guests on IBM System z Processors (z/VM-Planner) is a PC-based productivity tool that runs under MicroSoft Windows. It is designed to provide capacity planning insight for IBM mainframe processors running various Linux workload environments as guests under VM. Input consists primarily of z/VM guest definitions and capacity requirements for each intended Linux guest. The total guest capacity requirement is calculated based on the guest definitions and, since it is probable that all guests do not have peak utilization at the same time, a user-specified complementary peak percentage is used to calculate the true combined capacity requirements of the z/VM guests being sized. The tool also models z/VM processor requirements when merging new guests into existing VM images. The resulting guest capacity requirement is combined with that of VM to support the entire complement of guests.

The z/VM-Planner produces several graphs that depict the modeled system. The graphs illustrate the size of processor needed to support the modeled workload, the percent of that processor used by each application in the workload, and the estimated minimum and maximum processing capacity needed for the workload. Figure 7-6 on page 345 represents the minimum and maximum capacity needs for a modeled workload with five Linux guests running a mixture of Web applications, databases, and file servers.

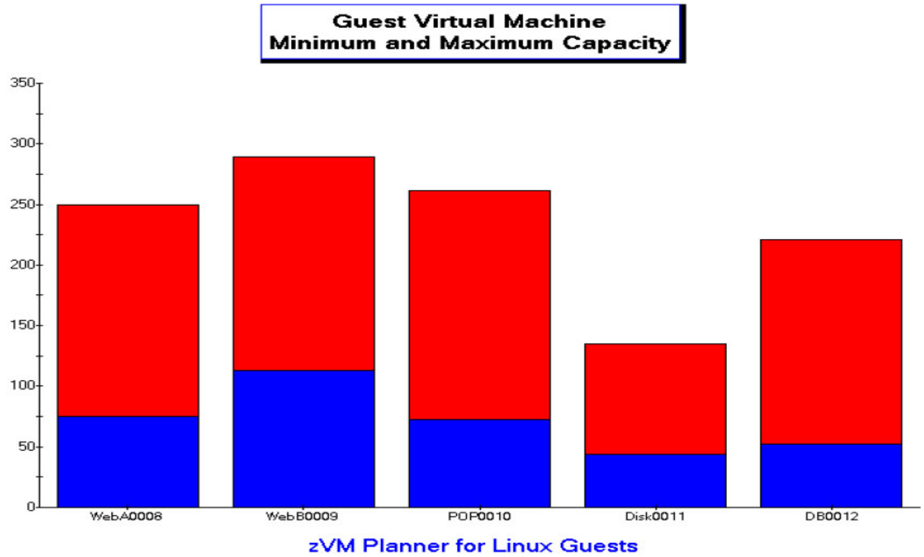

Figure 7-6 IBM z/VM Planner for Linux Guests

7.3.5 Tivoli Workload Scheduling Suite

The Tivoli Workload Scheduling Suite is the state-of-the-art production workload manager, designed to help you meet your present and future data processing challenges. Its scope encompasses your entire enterprise information system, including heterogeneous environments.

The Tivoli Workload Scheduling Suite simplifies systems management across heterogeneous environments by integrating systems management functions. There are three main components to the suite:

- Tivoli Workload Scheduler for z/OS: The scheduler in z/OS environments
- Tivoli Workload Scheduler: The scheduler in distributed environments
- Tivoli Job Scheduling Console. The common user interface for both Tivoli Workload Scheduler for z/OS and Tivoli Workload Scheduler.

> **Note:** For more information on Tivoli Scheduling Suite, refer to:
>
> http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?toc=/com.ibm.tivoli.itws.doc/toc.xml

Sizing considerations for Linux Guests

The Linux memory model has profound implications for Linux guests running under z/VM.

- z/VM memory is a shared resource.

Although aggressive caching reduces the likelihood of disk I/O in favor of memory access, the cost of caching must be considered. Cached pages in a Linux guest reduce the number of z/VM pages available to other z/VM guests.

- A large virtual memory address space requires more Linux kernel memory.

A larger virtual memory address space requires more kernel memory for Linux memory management. When sizing the memory requirements for a Linux guest, choose the smallest memory footprint that has a minimal effect on the performance of that guest. To reduce the penalty of occasional swapping that might occur in a smaller virtual machine, use fast swap devices.

Even though a 512 MB server does not require all that memory, it will eventually appear to use it all. Its memory cost is four times that of the 128 MB server.

> **Note:** Additional information can be found in the Performance Report on the z/VM Web site at:
>
> http://www.ibm.com/servers/eserver/zseries/zvm/perf/docs/
>
> Additional z/VM performance tips are available on the z/VM Web site at:
>
> http://www.ibm.com/servers/eserver/zseries/zvm/perf/tips/

7.3.6 Tuning memory for z/VM Linux guests

Storage tuning is very useful in virtualized environments, especially when the Hipervisor can overcommit resources. z/VM is the most functionally capable software Hipervisor operating system in existence, and fine tuning in all of the key resource areas, such as memory, is recommended for best overall system performance.

To minimize the system overhead and reduce its foot print, it is often helpful to reduce operational machine size. There are many ways by which the servers' memory foot print can be reduced. A common and intuitive approach is to kill un-needed processes. You can also tune z/VM memory usage by reducing the virtual machine size to the smallest possible size. The optimal smallest size may require a number of trials and experimentation.

Note: For more information on sizing of Linux virtual memory, refer to *Linux on IBM System z: Performance Measurement and Tuning*, SG24-6926

Dividing larger servers into smaller, more specialized servers can prove to be effective. Smaller servers can be tailored to perform specific operations. However, this is at a cost as each server will require some memory to run its own operating system.

Using a VDISK with the DIAGNOSE access method as a swap device reduces the performance penalty of swapping. This recommendation is applicable only if you have a large amount of the real memory available on System z. Using virtual disk for swap, reduces the I/O generated by swap on real devices, but at the same time increases the total amount of storage used. This tip is useful when a high paging rate comes from Linux and z/VM. In this situation, it is possible to experience effects of double paging (z/VM pages out guest storage already swapped out by Linux) and the use of VDISK as a swap device can help to reduce it.

Use the DIAGNOSE driver for the DASD and MDC record cache to reduce infrastructure storage costs. Using a record-level minidisk cache reduces the amount of storage required for MDC. This requires the DIAGNOSE driver.

Note: For more information refer to *IBM Linux on System z - Device Drivers, Features, and Commands*, SC33-8289.

If you are running Linux under z/VM you can exploit the Cooperative Memory Management 1 (CMM1) functions, providing a better storage allocation between guests.

Another way to reduce memory usage is to use shared kernel or execute-in-place technology. These new technologies can be used to reduce the total amount of storage needed to manage Linux instances.

7.3.7 Execute-in-place technology

You can minimize memory requirements and improve your performance of virtual Linux using discontiguous saved segments (DCSS). In a virtualized environment, Linux images may need the same data at the same moment. This means that the same data may get loaded into memory several times. A major part of memory required by Linux is used for binary application files and for shared library files. This technology helps you provide a way to share memory for some application using a DCSS managed by z/VM.

Considerations for data sharing:

- Application files can only be shared by Linux instances that use the same version of an application.
- Shared data must be read-only.
- Applications and libraries that are frequently used by numerous Linux instances are good candidates.

DCSS can be defined above Linux guest storage or in a storage gap. Usually, using DCSS in a storage gap is the preferred placement for a 64-bit Linux guest. Linux can access a DCSS through the execute-in-place file system (xip2).

Linux loads shared library and application files through the mmap() operation. mmap() maps the contents of a file into the application's address space. The xipl file system performs this operation by mapping the contents of the DCSS into the application's address space while other file systems use a page cache. This feature is called execute-in-place because the file contents are not physically copied to a different memory location.

With execute-in-place technology, Linux can:

- Access files and libraries without I/O operations (which increases overall performance).
- Run an application directly in a shared memory segment (which saves memory).

7.4 Monitoring your Linux Guests

Performance monitoring of Linux guests involves elaborate tasks of checking both your system z performance, as well as the guest operating systems. In the previous sections, we introduced scheduling concepts, as well as some important monitoring commands on system z/VM. This section addresses the Linux guest monitoring tasks.

The VMSTAT command

The **vmstat** command displays current statistics for processes, usage of real and virtual memory, paging, block I/O, and CPU. The left most columns show the number of running processes and number of blocked processes. The memory section shows memory being swapped out, free memory, the amount of buffer containing inodes and file metadata, and cached memory for files being read from disk. The swap section lists swaps in and swaps out. The I/O section reports the number (in kilobytes) of blocks read in and written out. System in and cs represent interrupts and context switches per second. The CPU section headers of us, sy, id, and wa represent percentage of CPU time count on users, system, idle, I/O wait, and steal, respectively. The vmstat command is useful in identifying memory shortages and I/O wait issues, in addition to situations where virtual CPUs are not backed up by physical CPUs.

Example 7-18 on page 349 lists system information after running vmstat with ten updates, five seconds apart, using the command:

vmstat 5 10

Example 7-18 .vmstat command

```
procs                    memory           swap      io       system     cpu
r  b  w   swpd   free  buff  cache  si  so   bi   bo    in    cs us  sy  id
0  0  0  29232 116972  4524 244900   0   0    0    0     0     0 0    0   0
0  0  0  29232 116972  4524 244900   0   0    0    0 2560     6 0    1  99
0  0  0  29232 116972  4524 244900   0   0    0    0 2574    10 0    2  98
```

pstree command

'pstree' shows the dependencies and relationships of applications. Similar to 'ps -eafx'.

Example 7-19 on page 349 illustrates the pstree command.

Example 7-19 pstree command

```
user@pserver1:~> pstree
   ...
   |-4*[nfsd]
   |-nmbd
   |-nscd---nscd---5*[nscd]
```

```
|-ntpd
|-portmap
|-rpc.mountd
|-rpc.statd
|-safe_mysqld---mysqld---mysqld---mysqld
|-2*[screen---bash]
|-scsi_eh_0
|-scsi_eh_1
|-smbd
|-sshd-+-2*[sshd---sshd---bash---vim]
|      |-sshd---sshd---bash---su---bash
|      `-sshd---sshd---bash---pstree
`-syslogd
```

top Command

'top' prints out system data per process. It shows an overview on the currently running system processes. The top command itself consumes CPU resources. Top Command is illustrated in Example 7-20 on page 350

Example 7-20 top command

```
[user@pserver1]$ top
1:40pm  up  3:17, 11 users,  load average: 0.00, 0.02, 0.03
134 processes: 127 sleeping, 6 running, 1 zombie, 0 stopped
CPU states:  1.9% user,  1.7% system,  0.0% nice, 96.2% idle
Mem:  1030464K av,  675028K used,  355436K free,       0K shrd,  156256K buff
Swap: 1663160K av,       0K used, 1663160K free                296068K cached
```

PID USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND

1746 wolf 15 0 54772 14M 11600 R 1.1 1.4 0:00 kdeinit

6147 wolf 15 0 1080 1080 840 R 0.9 0.1 0:00 topPrint the output

of the 'top' command to a file:

top d 4 b i > out

d refresh rate in seconds

b batch

i only active processes

System status (sysstat) tool

This package consists of several Linux tools to collect system data. The sysstat package is a widely used Linux standard tool. It is included in your distribution (RHEL 5 or SLES 10) or can be downloaded from the Web:

http://pagesperso-orange.fr/sebastien.godard/

If you install the source package, you have to compile it on Linux on System z to use it. The sysstat package consists of the following components:

- sadc data gatherer - stores data in binary file
- sar reporting tool - reads binary file created with 'sadc' and converts it to readable output
- mpstat - processor utilization
- iostat - I/O utilization

You can use *sar*, which prints the following information:

- Process creation
- Context switching
- All/single CPU utilization
- Network utilization
- Disk statistics

For a more detailed description see:

http://www.ibm.com/developerworks/linux/linux390/perf/tuning_how_tools.html#sysstat

OProfile tool

OProfile is an open source profiler for Linux systems. It offers profiling of all running code on a Linux system, including the kernel, shared libraries, application binaries, and so on, and provides a variety of statistics at a low overhead (varying from 1–8%) depending on the workload. It is released under the GNU GPL. OProfiler consists of a kernel driver, a daemon for collecting sample data, and tools for report generation.

OProfile generally supports Linux kernel 2.2, 2.4, 2.6, and later.

> **Note:** If you are running:
>
> - SUSE Linux, the kernel level must be at least kernel-s390(x)-2.6.5-7.191, which starts from SLES9 SP2.
> - RedHat Linux, the kernel level must be at least kernel-2.6.9-22.EL, which is RHEL4 U2.
> - Linux on System z, OProfile only supports kernel 2.6 or later.

OProfile utilizes CPU's performance counters to count events for all of the running code, and aggregates the information into profiles for each binary image. However, System z hardware currently does not have support for this kind of hardware performance counters utilized by OProfile, so the timer interrupt is used instead.

Example 7-21 on page 352 illustrates a typical usage of OProfile on a Linux guest running on z/VM. To use the OProfile tool, the timer should be turned on by using command *sysctl*. After all the profiling is done, turn the timer off. To set up the environment, use the *opcontrol* command to tell OProfile the location of the vmlinux file corresponding to the running kernel. Then it is ready to collect the profile data. After the test and tool shutdown, the profiling report can be generated.

Example 7-21 Oprofile tool

```
sysctl -w kernel.hz_timer=1 [1]

gunzip /boot/vmlinux-2.6.5-7.201-s390x.gz [2]

opcontrol --vmlinux=/boot/vmlinux-2.6.5-7.201-s390x [3]
```

opcontrol --start [4]

26 Linux on IBM System z: Performance Measurement and Tuning

<DO THE TEST>

opcontrol --shutdown [5]

opreport [6]

Note the following points:

[1] The command to turn on the timer. When the profiling is finished, specify it to be 0 to turn off the timer.

[2] Unzip the vmlinux file if none exists.

[3] Specify the vmlinux file for the running kernel.

[4] Start the OProfile deamon, and begin profiling.

[5] Stop the OProfile deamon, and stop profiling.

[6] Product symbol or binary image summaries.

Note that in Figure 7-7 on page 353, the CPU information is not correctly displayed. The System z hardware does not support the CPU performance counters required by OProfile yet, however the output does contain some valuable information.

```
CPU: CPU with timer interrupt, speed 0 MHz (estimated)
Profiling through timer interrupt
vma         samples %        app name          symbol name
80002840    5862     34.8970 mcf_base.z_Linux  price_out_impl
800012c8    5221     31.0811 mcf_base.z_Linux  refresh_potential
80003cb4    4398     26.1817 mcf_base.z_Linux  primal_bea_mpp
80003b60    408      2.4289  mcf_base.z_Linux  sort_basket
0001a67c    345      2.0538  vmlinux           default_idle
800013d8    138      0.8215  mcf_base.z_Linux  flow_cost
800033bc    98       0.5834  mcf_base.z_Linux  update_tree
800020f8    88       0.5239  mcf_base.z_Linux  dual_feasible
800036a4    72       0.4286  mcf_base.z_Linux  primal_iminus
8000323c    40       0.2381  mcf_base.z_Linux  write_circulations
80002720    24       0.1429  mcf_base.z_Linux  insert_new_arc
```

Figure 7-7 OProfile output

The output includes:

- vma (Virtual Memory Area): a contiguous area of virtual address space. These areas are created during the life of the process when the program attempts to memory map a file, links to a shared memory segment, or allocates heap space.
- samples: the number of samples for the symbol.
- “%”: the percentage of samples for this symbol relative to the overall samples for theexecutable.
- app name: the application name to which the symbol belongs.
- symbol name: name of the symbol that was executed.

From the report of OProfile, we can analyze the performance of a target module and adjust tuning accordingly. For example, in Figure 7-7 on page 353, modules of the application mcf_bas.z_Linux consume most of the CPU time. To improve the performance of this application, more attention should be paid to these modules than the others.

> **Note:** For more detail and documentation of the OProfile commands, refer to:
>
> http://oprofile.sourceforge.net/docs/

8

System events and logs, backup and recovery

Linux on System z is a reliable environment to run applications. However,error situations could still occur. For instance, you may encounter:

- software errors: abnormal program termination, invalid input data
- operating system errors: at the Linux or z/VM level
- hardware errors: LPAR, system or disk subsystem failure.

This chapter provides an overview of some of the tools available in z/VM and Linux to collect and analyze system events, and to create backups and restore them in case of an unrecoverable failure.

Objectives: Upon completion of this chapter, you should be familiar with:

- some tools available for system events and errors logging
- some tools available for errors analysis
- some of the various options that can be considered for backing up and restoring data.

8.1 z/VM and Linux errors handling

This section gives the reader an overview of how errors are handled in z/VM and Linux, and some details about some error codes.

8.1.1 z/VM default error handling

Each z/VM feature, module or facility, is responsible for handling the “user-related” errors that occur when a command is issued. These errors can be due - but are not limited - to incorrect configuration, wrong invocation of a command, irrelevant input data. When such an error occurs, the function that fails informs the user with an appropriate error code and message.

For instance, when trying to log on to a virtual machine that is not defined in z/VM, CP issues the message shown in Example 8-1

Example 8-1 Error message when logging on to an undefined virtual machine

```
L LASMAYOU LASMAYOU
HCPLGA053E LASMAYOU not in CP directory

Enter one of the following commands:

  LOGON userid           (Example:  LOGON VMUSER1)
  DIAL userid            (Example:  DIAL VMUSER2)
  MSG userid message     (Example:  MSG VMUSER2 GOOD MORNING)
   LOGOFF
```

For more details about z/VM error codes and messages, refer to Chapter 8.1.2, “z/VM error codes” on page 357

System related errors - intermittent and permanent machine errors, system I/O errors - are handled automatically by CP, in a way that is completely transparent to the end-user. However, when an error occurs, CP records it and sends a message to your z/VM primary system console. The message notifies you of the error and tells you whether:

- System operation can continue
- System operation can continue, but with fewer resources
- System restart and recovery is beginning.

If the error occured in an identified virtual machine, and provided that the overall z/VM system’s integrity is preserved, a soft abend occurs. During a soft abend, CP takes a soft abend dump, and operations continue.

If the error is more critical, CP terminates in a hard abend. During the abend, CP:

- saves z/VM spool files and system data file queues
- takes a hard abend dump, that can be used later for debugging purposes
- tries to restart itself and the operator's console virtual machine.

> **Note:** All guest operating systems that were running before the abend have to be restarted manually - unless the appropriate IPL commands have been set up.

In some cases, CP will not be able to restart, and will then inform you of the required steps to recover from the error.

8.1.2 z/VM error codes

Most of z/VM error codes follow the same format. They are made of a message identifier, and a message text.

The message identifier format is the following:

xxxmmm###s or xxxmmm####s.

- First 3 characters are the component identifier. See Table 8-1 on page 357 for a list of the major components identifiers
- Next 3 characters are a code that identifies, in a given component, the module or function that fails
- Following 3 or 4 digits are the message number itself
- The last digit is used as a severity code:
  - A: immediate action is required
  - D: Decision
  - E: Error
  - I: Information only
  - R: Response
  - S: Severe Error
  - T: Terminating Error
  - W: System Wait (CP only), warning (all others)

Table 8-1 z/VM major components identifiers

Prefix	z/VM component
DMS	CMS (Conversational Monitoring System)
DTC	TCP/IP for z/VM

Prefix	z/VM component
DVH	Directory Maintenance Facility for z/VM
FCX	Performance Toolkit for z/VM
HCP	CP (Control Program)
RAC	RACF Security server
TCP	TCP/IP for z/VM

This list is not exhaustive. For more information, please see the *z/VM: CP Messages and Codes, GC24-6119* (http://publib.boulder.ibm.com/infocenter/zvm/v5r3/topic/com.ibm.zvm.v53.hcpw0/msgfrmt.htm#msgfrmt)

To find an explanation of the error messages, and some hints to solve the problem, it is possible to use the HELP command.

For instance, to find an explanation for the error message shown in Example 8-1, you can type the following command: **help HCPLGA053E**:

The result is shown in Example 8-2:

Example 8-2 HCPLGA053E error message help panel

```
MSG HCPLGA053E                All Help Information                  line
1 of 12
(c) Copyright IBM Corporation 1990, 2007

 HCP053E   (XAUTOLOG failed for userid:) <userid|value> not in CP
directory

 Explanation:  The user ID supplied was not found in the z/VM
directory. If the command was an asynchronous XAUTOLOG, the message
indicates which user ID did not get logged on.

 System Action:  The command is not executed; system operation
continues.

 User Response:  Reissue the command with a valid user ID.

* * * End of File * * *
```

Note: In the HELP command, it is possible to omit the 3-digits module code. The command HELP HCP053E gives the same result.

For more details about the HELP command, refer to Chapter 2.5.1, "Using the help command in CMS" on page 40.

8.1.3 Linux error handling

Inside a Linux virtual machine, it is the Linux kernel that is responsible for the correct resources allocation and isolation of the processes that run. The kernel tries to recover from errors as transparently as possible for applications and end-users.

Most of the errors are application errors, bad programming practices or invalid input data for instance. In case an application tries to allocate resources it is not allowed to use, for instance write to a memory page already in use by another application, the Linux kernel issues a segmentation fault, also know as segfault or SIGSEGV, and the application ends abnormally. A coredump file is created for further analysis of the error in the application. This ensures an application doesn't interfere with another.

When an unrecoverable error occurs, the Linux kernel goes into "kernel panic" mode. It writes a error message on the console, saves an image of the memory on the disk for post-mortem debugging, and reboots. Classical causes of kernel panics are attempts to read or write an invalid or restricted memory address. Sometimes, hardware failures can also result in kernel panics.

8.1.4 Linux error codes

As opposed to z/VM, there's no pre-defined error messages format in Linux. Most of the error messages are made of one or more strings that identify the component that encountered the error - either the kernel itself or a daemon - and a message giving the cause of the error. See for instance Example 8-14 on page 374 for an error message from the kernel module qeth.

In some cases, the return code of the application can also be considered as the error message. Return codes different from zero indicate something went wrong in the application.

Note: Even though Linux provides some predefined errors and return codes in its standard library, each developer is free to define its own error codes, different from the standard ones.

8.2 z/VM tools for error gathering and analysis

As a z/VM system operator, your job is to ensure that all the resources available to the z/VM LPAR are available to the users to get their job done. Part of system operator's job includes collecting information about:

- Accounting
- System performance
- System events
- Error recording

A set of defaults users are available in z/VM to accomplish those tasks:

- DISKACCNT collects accounting and billing information.
- OPERSYMP records symptoms records.
- OPERATOR logs system events
- EREP records error records.

To check the status of the various recordings on a z/VM system, the command **query recording** (abbreviated as **q rec**) can be used, as shown in Example 8-3.

Example 8-3 Output of the query rec command

```
q rec
RECORDING   COUNT    LMT USERID   COMMUNICATION
EREP     ON  00000000  002 EREP     ACTIVE
ACCOUNT  ON  00000000  020 DISKACNT ACTIVE
SYMPTOM  ON  00000000  002 OPERSYMP ACTIVE
Ready; T=0.01/0.01 14:58:23
```

This chapter focuses on collecting systems events and error recording.

8.2.1 Collecting information about system events

When z/VM IPLs, the system operator machine (OPERATOR by default) is automatically logged on; if the NOAUTOLOG parameter has been specified at IPL time, no other machines are started. On the OPERATOR console, all system related and error messages are displayed. All commands issued by the operator, as well as the answers to these commands, are also displayed on the OPERATOR console.

The system operator machine is defined in the z/VM configuration file, SYSTEM CONFIG, as shown in Example 8-4.

Example 8-4 System Userids configuration

```
/*******************************************/
/*                            System Userids */
/*******************************************/

  System_Userids   ,
    Operator  OPERATOR  ,
    Account   DISKACNT  ,
    Dump      OPERATNS  ,
    Erep      EREP
```

This console log can be saved into a spool file, to be able to analyze it later. In case of a system abend, CP also saves by itself the console log into a spool file.

Status of console logging

To find out whether or not the console activity is logged into a file, issue the command **query virtual console** (abbreviated as **q v console**, or **q v con**), as explained in Example 8-5:

Example 8-5 Status of console logging

```
q v console
00: CONS 0009 ON LDEV L0003   TERM STOP  HOST TCPIP    FROM
9.57.138.243
00:     0009 CL T NOCONT NOHOLD COPY 001    READY FORM STANDARD
00:     0009 TO LNXGUI   PRT DIST LNXGUI    FLASHC 000 DEST OFF
00:     0009 FLASH       CHAR      MDFY       0 FCB      LPP OFF
00:     0009 3215   NOEOF CLOSED    NOKEEP NOMSG NONAME
00:     0009 SUBCHANNEL = 0008
Ready; T=0.01/0.01 15:28:01
```

The first line of the output says STOP, meaning console logging is not enabled.

Enabling console logging

To enable console logging in a virtual machine, say LNXGUI, use the command **cp spool console start** (abbreviated as cp sp con start), as shown in Example 8-6 on page 362. Check the status again with **query virtual console**. The first line of the output should say START, meaning console logging is started.

Example 8-6 Enabling console logging

```
q v console
00: CONS 0009 ON LDEV L0003   TERM STOP  HOST TCPIP    FROM
9.57.138.243
00:     0009 CL T NOCONT NOHOLD COPY 001   READY FORM STANDARD
00:     0009 TO LNXGUI  PRT DIST LNXGUI   FLASHC 000 DEST OFF
00:     0009 FLASH      CHAR      MDFY      0 FCB      LPP OFF
00:     0009 3215  NOEOF CLOSED   NOKEEP NOMSG NONAME
00:     0009 SUBCHANNEL = 0008
Ready; T=0.01/0.01 15:28:01
cp spool console start
Ready; T=0.01/0.01 15:31:13
q v console
00: CONS 0009 ON LDEV L0003   TERM START HOST TCPIP    FROM
9.57.138.243
00:     0009 CL T NOCONT NOHOLD COPY 001   READY FORM STANDARD
00:     0009 TO LNXGUI  PRT DIST LNXGUI   FLASHC 000 DEST OFF
00:     0009 FLASH      CHAR      MDFY      0 FCB      LPP OFF
00:     0009 3215  NOEOF OPEN 0039 NOKEEP NOMSG NONAME
00:     0009 SUBCHANNEL = 0008
Ready; T=0.01/0.01 15:31:19
```

In Example 8-7, we ensure console logging is activated next time the virtual machine IPLs, by adding the statement **cp spool console start** into the LNGXGUI machine PROFILE EXEC.

Example 8-7 Updating PROFILE EXEC

```
PROFILE  EXEC    A1  V 130  Trunc=130 Size=3 Line=0 Col=1 Alt=2

===== * * * Top of File * * *

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7.
..
===== /* */
===== 'CP SET PF12 RETR'
===== 'CP SPOOL CONSOLE START'
===== * * * End of File * * *

====>
                                                             X E D I T
1 File
```

Saving console log to a file

To save the console log to a file, it is necessary to stop console logging. A file will be created into the machine's virtual printer. This file can either be printed directly, or sent to the virtual reader for further processing. Example 8-8 shows the commands used for this purpose.

Example 8-8 Saving console log to a file

```
cp sp console close
00: PRT FILE 0041 SENT FROM LNXGUI   CON WAS 0041 RECS 0131 CPY  001 T
NOHOLD NO
KEEP
Ready; T=0.01/0.01 16:47:06
change prt all to * rdr
00: RDR FILE 0041 SENT FROM LNXGUI   PRT WAS 0041 RECS 0131 CPY  001 T
NOHOLD NO
KEEP
00: 0000001 FILE  CHANGED
Ready; T=0.01/0.01 16:47:16
```

The command **cp spool console close** (abbreviated as **cp sp console close**, or cp sp con close) closes the current log file, and opens another one. The current log file is sent to the virtual machine printer. This file has spoolid 0041. The **change printer all to * readerr** (abbreviated in the example as **ch prt all to * rdr**) command is used to move all files from the machine virtual printer to its virtual reader. The asterisk "*" means "myself", the virtual machine in which the command is issued.

Note: It is also possible to close to spool and have it sent to your reader in one single command:

cp sp console close *

To send only one specific file from the printer to the reader of a virtual machine, it is possible to specify the spoolid of the file to transfer:

ch prt 0041 to * rdr

Once in the reader, the files can be listed using the **RL** command and browsed (using the **PEEK *spoolid*** command, that is mapped to the F11 key in the reader's list window) as shown in Example 8-9 on page 364, saved to disk and archived.

Example 8-9 Displaying a console log file

```
0041    PEEK    A0  V 133  Trunc=133 Size=131 Line=52 Col=1 Alt=0
File (none) (none) from LNXGUI at VMLINUX6 Format is CONSOLE.
z/VM V5.3.0   2008-05-06 13:40

Ready; T=0.01/0.01 16:45:57
q da all
00: HCPQVC022E A virtual device number was not supplied or it was
invalid.
Ready(00022); T=0.01/0.01 16:46:04
q da
00: DASD 0101 9336 (VDSK) R/W    200000 BLK ON DASD  VDSK SUBCHANNEL =
0001
00: DASD 0190 3390 LX6RES R/O       107 CYL ON DASD  1A20 SUBCHANNEL =
0009
00: DASD 0191 3390 LX6W02 R/W        10 CYL ON DASD  1A24 SUBCHANNEL =
0000
00: DASD 019D 3390 LX6W01 R/O       146 CYL ON DASD  1A23 SUBCHANNEL =
000A
00: DASD 019E 3390 LX6W01 R/O       250 CYL ON DASD  1A23 SUBCHANNEL =
000B
00: DASD 0401 3390 LX6W01 R/O       146 CYL ON DASD  1A23 SUBCHANNEL =
000E
00: DASD 0402 3390 LX6W01 R/O       146 CYL ON DASD  1A23 SUBCHANNEL =
000D
00: DASD 0405 3390 LX6W01 R/O       156 CYL ON DASD  1A23 SUBCHANNEL =
000F
00: DASD 0592 3390 LX6W01 R/O        70 CYL ON DASD  1A23 SUBCHANNEL =
000C
Ready; T=0.01/0.01 16:46:06
link * 100 100 MR
Ready; T=0.01/0.01 16:46:15
link * 150 150 MR
Ready; T=0.01/0.01 16:46:20
cp console close
Unknown CP command
Ready(-0001); T=0.01/0.01 16:46:51
q v
00: STORAGE = 512M
1= Help      2= Add line  3= Quit      4= Tab      5= Clocate     6=
?/Change
7= Backward  8= Forward   9= Receive  10= Rgtleft 11= Spltjoin   12=
Cursor
```

```
====>
                                                              X E D I T
1 File
```

This console log file shows all the commands that have been issued in the console of the virtual machine, and the answer to these commands. In case CP encounters an error, this error will be logged in the console log as well.

> **z/OS analogy:** This is similar to z/OS SYSLOG.

It is also possible to use a virtual machine's console to display several users' consoles. This is done by using the **SET OBSERVER** command, or by adding a parameter in the user directory CONSOLE statement. See Example 8-10 below for an example of a CONSOLE directory statement.

Example 8-10 LNXGUI virtual machine directory CONSOLE statement

```
USER    DIRECT   C1  F 80  Trunc=72 Size=2041 Line=60 Col=1 Alt=1

00060   SPOOL 000E 1403 A
00061    CONSOLE 009 3215 T LNXMAINT OBSERVER
00062   LINK MAINT 0190 0190 RR
00063   LINK MAINT 019D 019D RR
00064    LINK MAINT 019E 019E RR
```

Saving LNXMAINT machine console to a file allows the operator to analyze the messages issued from several virtual machines, as shown in Example 8-11, where LNXMAINT virtual machine is OBSERVER for LNXGUI, LNXCER and LNXKEN.

Example 8-11 LNXMAINT console output

```
CP DISC
LNXGUI  : 00: DISCONNECT AT 15:59:53 EDT THURSDAY 06/05/08
LNXGUI  : 00:
Press enter or clear key to continue
LNXKEN  : 00: z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
built on IBM Virtualization Technology
LNXKEN  : 00: There is no logmsg data
LNXKEN  : 00: FILES: 0003 RDR,   NO PRT,   NO PUN
LNXKEN  : 00: RECONNECTED AT 16:00:02 EDT THURSDAY 06/05/08
LNXKEN  : 0
LNXKEN  : 0
LNXKEN  : Password:
LNXCER  : 00: z/VM Version 5 Release 3.0, Service Level 0701 (64-bit),
```

```
built on IBM Virtualization Technology
LNXCER  : 00: There is no logmsg data
LNXCER  : 00: FILES: 0003 RDR,   NO PRT, 0001 PUN
LNXCER  : 00: RECONNECTED AT 16:00:18 EDT THURSDAY 06/05/08
```

Note: Refrain from using the OPERATOR console as an observer/user for other machines. OPERATOR should be reserved for system related messages only.

8.2.2 Collecting information about errors

By default, when z/VM is running, error data collection is turned on by CP.

CP generates an error record:

- each time a hardware problem occurs in the LPAR while CP is active.
- each time a hardware problem occurs in a virtual machine in supervisor state.

Note: Error recording doesn't occur in virtual machine in emulation state, unless the user has issued CP SET SVC76 command in this virtual machine.

z/VM error records analysis is a 3-steps process:

1. When a hardware error occurs, CP generates an error record, and notifies the error recording virtual machine.
2. The error recording virtual machine is in charge of retrieving the error records from storage and saving them to disk.
3. Last step is to process the records saved to disk, to generate the reports that will be used by system operators or support teams to identify problems.

Environmental Record Editing and Printing Program - EREP - is the application (virtual machine) dedicated to error records processing in z/VM, but also in z/OS and z/VSE.

Generating EREP reports

EREP functions can be used to create many types of reports:

- System Summary Report

provides an overview of errors for each component of the environment: CPU, storage, channels, I/O subsystem.

- System Exception Report

  lists software and hardware error data in a variety of ways.

- Threshold Summary Report

  shows all the permanent read/write errors, temporary read/write errors, and media statistics for each volume mounted, for 3410, 3420, and 8809 tape devices.

- CCH and MCH Detail Report

  reports Channel CHecks and Machine CHecks.

- MDR and OBR Detail Report for Controllers

  reports 3704, 3705, 3720, 3725 and 3745 tape controllers related errors.

- Detail Summaries for I/O Errors
- Detail Reports for Software Records
- Trends Report

  presents the pattern and frequency of errors on a daily basis. You can use these reports to see when the errors began, their pattern, and when they end.

- Event History Report

  shows errors in a time sequence that allows you to see how often and in what order errors occur.

EREP reports are generated by the CPEREPXA class F command, with a defined set of arguments. To generate an EREP report under z/VM, it is necessary to define the input and output files using FILEDEFs and then invoke the CPEREPXA EXEC. CPEREPXA and EREP operands can be entered either as a parameter file or at CPEREPXA prompt.

Note: EREP virtual machine stores error records on its 191 disk, in files named XAEREPIO RECORD and XAREPMC RECORD.

z/OS analogy: To create EREP reports under z/OS, it is necessary to define the input and output data sets using JCL DD statements. The JCL submits the job as a batch job or interactively via TSO. Put the IFCEREP1 program in the JCL EXEC statement. Include the EREP parameters on the EXEC statement or as part of SYSIN in-stream data with the EREP control statements.

Detailing all configuration options of CPEREPXA is far beyond the scope of this book. Refer to the *EREP V3R5 User's Guide, GC35-0151* (http://publib.boulder.ibm.com/infocenter/zvm/v5r3/topic/com.ibm.zos.r9.ifc1000/ifc5g102.htm) for complete reference and detailed options

Example 8-12 is an example of a System Summary Report as generated by CPEREPXA.

Example 8-12 Example of System Summary Report

```
SYSTEM   SUM     A1  F 132  Trunc=132 Size=95 Line=2 Col=1 Alt=0
====>
 S Y S T E M  S U M M A R Y           REPORT DATE 141 08
         (PART 1)                     PERIOD FROM 065 08
 CPU/CHANNEL/STORAGE/SCP                         TO 115 08

                     TOTAL CPU-0
 IPL                     54    54

 MACHINE CHECK

   RECOVERABLE            0     0
   NON-RECOVERABLE        0     0

 CHANNEL CHECK

   CHANNEL 0              0     0
   CHANNEL 1              0     0
   CHANNEL 2              0     0
   CHANNEL 3              0     0
   CHANNEL 4              0     0
   CHANNEL 5              0     0
   CHANNEL 6              0     0
   CHANNEL 7              0     0
   CHANNEL 8              0     0
   CHANNEL 9              0     0
   CHANNEL A              0     0
   CHANNEL B              0     0
   CHANNEL C              0     0
   CHANNEL D              0     0
   CHANNEL E              0     0
   CHANNEL F              0     0

 PROGRAM ERROR

  ABEND                 344   344
```

```
 PRGM INT              21    21

END OF DAY              0     0

 CPU  MODEL  SERIAL NO.
 0   2084XA 0BE34E

S Y S T E M  S U M M A R Y             REPORT DATE 141 08
        (PART 1 CONTINUED)             PERIOD FROM 065 08
SUBCHANNEL/CHANNEL                              TO 115 08

                   TOTAL CPU-0

SUBCHANNEL LOGOUT

CHANNEL REPORT WORD

 HARDWARE               0     0
 SOFTWARE               0     0

TOTAL RECORDS         419   419

CPU  MODEL  SERIAL NO.
 0   2084XA 0BE34E

S Y S T E M  S U M M A R Y             REPORT DATE 141 08
         (PART 2)                     PERIOD FROM 065 08
      I/O SUBSYSTEM                             TO 115 08

              ----- TOTAL ----    CPU-0
               PERM  TEMP  PATH  PERM TEMP
DASD        ******************************

2107-SSID 8A00    0     3     0     -    -
2107-SSID 8C00    0     3     0     -    -
2107-SSID 8000    0     5     0     -    -
2107-SSID 8400    0     7     0     -    -
2107-SSID 8600    0    14     0     -    -
2107-SSID 8700    0     2     0     -    -
2107-SSID 8800    0     4     0     -    -

TAPE        ******************************

 3590 FA50         0      1      0       0    1
3590 FA51        0     1     0      0    1
```

```
3590 FB00        0     1     0      0    1
3590 FB01        0     1     0      0    1
3590 FB02        0     1     0      0    1
3590 FB03        0     1     0      0    1
3590 FB10        0     1     0      0    1
3590 FB11        0     1     0      0    1
3590 FB12        0     1     0      0    1
3590 FB13        0     1     0      0    1
3590 FB30        0     2     0      0    2
3590 FB40        1     6     0      1    6
3590 FB41        0     7     0      0    7

TOTALS           1    63     0      1   25
CPU  MODEL  SERIAL NO.
 0   2084XA 0BE34E
 * * * End of File * * *
```

8.2.3 Dumps and traces

This section gives an overview of the dumps and traces facility in z/VM.

Dumps

A dump is a picture of either the virtual machine or the partition's storage. If a problem is encountered, it can probably be found somewhere in this picture. A dump can be used to identify the moment in time when malfunctions begin.

Several z/VM components can generate dumps:

- CP itself.
- A virtual machine in which CMS, or another z/VM component, or a guest operating system is running.
- A communication controller.

Several types of dumps are available in z/VM, depending on the component that created it, or the information required for analysis:

- A CP dump. This is a dump of the storage directly owned by CP. It is generated by CP during a hard abend and results in system termination and possibly a restart.

> **Note:** The **query dump** (abbreviated as **q dump**) command shows the device that will be used to store the CP DUMP file. CP DUMPs will be created in the virtual reader of the virtual machine defined in the SYSTEM CONFIG, under System_UserIDs - OPERATNS by default. See Example 8-4 on page 361 for an example SYSTEM CONFIG Dump statement.

- A snapdump. This is a dump of the storage directly owned by CP and is very similar to a hard abend dump but does not result in system termination.
- A CP soft abend dump. A soft abend dump is a dump of a small amount of the storage directly owned by CP. It is created when CP encounters a problem where system integrity is not compromised by the error, or when CP can isolate an error to a virtual machine. It does not result in system termination.
- A stand-alone dump. Sometimes, a problem can be so severe that your system cannot even produce a CP dump on its own. For this reason, every z/VM system is equipped with a special program that produces a dump of real storage, regardless of how severe the problem is. It is called a stand-alone dump because the program that produces it stands alone or independent of the rest of the system programming. Because it is independent of the system programming, any problems there will not prevent the dump from being created.
- A dump limited to any single virtual machine (VMDUMP) running in your z/VM system. For example, you can request a dump of a virtual machine containing CMS, RSCS, or any guest operating system that resides in a virtual machine.
- A dump of a communication controller's storage.

If your virtual machine is running a Guest operating system such as Linux or z/OS, it is recommended to use the tools available for that operating system rather than the z/VM dump tools. Refer to Chapter 8.3.2, “Error analysis tools” on page 377 for a discussion about Linux dump tools.

For more details about VM dump tools, please refer to z/VM V5R3.0 Diagnosis Guide, GC24-6092 (`http://publib.boulder.ibm.com/infocenter/zvm/v5r3/index.jsp?topic=/com.ibm.zvm.v53.hcpc1/getdump.htm`)

> **z/OS analogy:** in z/OS, a dedicated dump adress space is used to capture dump onto DASD.
>
> User-initiated dumps will also be saved in the dump address space.

Traces

The virtual machine tracing facility allows you to follow - or trace - the execution of almost anything that takes place while you are using your virtual machine. For example, you can trace instructions, I/O interrupts, branches, and changes to storage.

Tracing the execution of programs inside a virtual machine can help system operators and support teams to identify the step in a program where execution fails. See for instance the output of the **TRACE I/O** command run in a Linux guest in Example 8-13.

Example 8-13 Example of tracing I/O in a Linux guest running cat /root/.viminfo command.

```
00:  EXTENT              80C00000 0000000A 0008000D 0008000D
00: CCW   1F209FC0  47400010 1F209FF0   0008  47400010 ........
00:  LOCATE RECORD        01800001 0008000D 0007000D 01061000
00: CCW   1F209FC8  85001000 1DE19000   0010  85041000 ........
00:  IDAL                            00000000DAAED000
00:  -> 00000000208C790E'  SIGA  B2740000   0000000000000000   CC 0
00:            SCH 0006    DEV 0602
00:           FC  00000001    MASK    80000000    00010006
00:  -> 0000000000332258'  SSCH  B2333000   00000000010F4844   CC 0
00:            SCH 0001    DEV 0100
00:           CPA 1F209FB8   PARM 010F4800    KEY 0  FPI C2  LPM F0
00: VDEV 0100 CCW 63600020 1F209FD0
00: CCW   1F209FB8  63600020 1F209FD0   0000  63600020 ........
00:  EXTENT              80C00000 0000000A 0008000D 0008000D
00: CCW   1F209FC0  47400010 1F209FF0   0008  47400010 ........
00:  LOCATE RECORD        01800001 0008000D 0007000D 02181000
00: CCW   1F209FC8  85001000 1E1DE000   0010  85041000 ........
00:  IDAL                            000000005CCCB000
00:  -> 00000000208C8B52'  SIGA  B2740000   0000000000000000   CC 0
00:            SCH 0006    DEV 0602
00:           FC  00000000    MASK    20000000    20000000
00:  -> 00000000208C8B52'  SIGA  B2740000   0000000000000000   CC 0
00:            SCH 0006    DEV 0602
00:           FC  00000000    MASK    20000000    20000000
00:  -> 00000000208C8B52'  SIGA  B2740000   0000000000000000   CC 0
00:            SCH 0006    DEV 0602
00:              FC   00000000    MASK    20000000    20000000
```

For more information about z/VM trace command, please refer to z/VM V5R3.0 System Operation, SC24-6121 at

(http://publib.boulder.ibm.com/infocenter/zvm/v5r3/topic/com.ibm.zvm.v53.hcpb2/uc9a.htm#uc9a)

8.3 Linux activity logging and error analysis

Activity logging is built in every Linux distribution. Every kernel-related messages, every daemon related messages, is logged into a common directory, whatever the distribution is.

These messages can be either:

- information messages

  for instance: somebody has logged on the machine through ssh from a given IP adress.

- warning messages

  for instance: one of the configuration options of a daemon cannot be applied, but didn't prevent the daemon from starting

- error messages

  for instance: someone unplugged the network cable from your network interface.

Going through the contents of these log files is a first step for error identification.

8.3.1 Linux activity logging

Two Linux daemons responsible for logging kernel and user activities: klogd and syslog, or syslog-ng:

- klogd for kernel-related messages.

  klogd is the kernel logger. It intercepts and translates the messages issued by the kernel itself, and passes them in a formatted way to syslog for archiving

- syslog and its variants for everything in user space.

  syslog is the userspace logger. It collects information from various daemons running in the system, and saves the messages into files for further reference.

Log files of interest

Whatever the distribution is, log files are saved in a common directory, called /var/log. However, log files names can vary from one distribution to another.

> **Note:** /var/log is the standard location for the log messages of the daemons provided by the distribution packages. Third-party software will most of the time save their log files in different locations.

The most relevant files in /var/log directory, when looking for errors, are:

- On SuSE Linux Enterprise Server 10
  - *messages*, that includes all system messages

  In Example 8-14 on page 374, there is a report from the qeth driver informing that a network failure occurred at 16:41:21 on May, 21st.

Example 8-14 an extract of /var/log/messages log file on SLES10.

```
May 21 15:17:15 lnxguill kernel: audit(1211397428.415:2):
AppArmor (version 2.0-19.43r6320) initialized
May 21 15:17:15 lnxguill syslog-ng[1206]: Changing permissions on
special file /dev/xconsole
May 21 15:17:15 lnxguill syslog-ng[1206]: Changing permissions on
special file /dev/tty10
May 21 15:17:15 lnxguill kernel:
May 21 15:17:15 lnxguill kernel: Unable to find swap-space
signature
May 21 15:17:23 lnxguill kernel: eth0: no IPv6 routers present
May 21 15:18:06 lnxguill sshd[1325]: Accepted
keyboard-interactive/pam for root from 9.57.138.243 port 2805
ssh2
May 21 16:17:11 lnxguill syslog-ng[1206]: STATS: dropped 0
May 21 16:41:28 lnxguill kernel: qeth: Link failure on eth0
(CHPID 0x20) - there is a network problem or someone pulled the
cable or disabled the port.
May 21 16:41:28 lnxguill mingetty[1319]: ttyS0: invalid character
for login name found
May 21 16:42:05 lnxguill kernel: qeth: Link reestablished on eth0
(CHPID 0x20). Scheduling IP address reset.
May 21 16:42:05 lnxguill kernel: qeth: Recovery of device
0.0.0600 started ...
May 21 16:42:05 lnxguill kernel: qeth: Device
0.0.0600/0.0.0601/0.0.0602 is a Guest LAN QDIO card (level: V530)
May 21 16:42:05 lnxguill kernel: with link type GuestLAN QDIO
(portname: )
May 21 16:42:05 lnxguill kernel: qeth: Hardware IP fragmentation
not supported on eth0
```

  - *lastlog*, used to see the last logged on users. Use the command lastlog to query this file
  - *faillog*, used to see the users that failed to log on. Use the command faillog to query this file
  - *warn* contains warning information - such as access denied from ssh for instance.
- On RedHat Enterprise Linux 5
  - *messages*, includes all system messages.
  - *lastlog*, used to see the last logged on users
  - *faillog*, used to see the users that failed to log on.
  - *secure* for all security related messages.

Browsing log files

Lots of commands are available to browse log files.

The most standard way to browse a log file is to open it using a text editor. Another convenient way to display log files is to use the **tail** command, that outputs the last lines of the files. The output is refreshed each time a new time is written in the file. This command allows live log monitoring, so it’s a convenient way to see what’s written to the files when trying to reproduce an error.

Example 8-15 Output of tail -f /var/log/messages

```
lnxguill:~ # tail -f /var/log/messages
May 29 13:54:30 lnxguill kernel: AppArmor: AppArmor (version
2.0-19.43r6320) initialized
May 29 13:54:30 lnxguill kernel: audit(1212083662.366:2): AppArmor
(version 2.0-19.43r6320) initialized
May 29 13:54:30 lnxguill syslog-ng[1209]: Changing permissions on
special file /dev/xconsole
May 29 13:54:30 lnxguill syslog-ng[1209]: Changing permissions on
special file /dev/tty10
May 29 13:54:30 lnxguill kernel:
May 29 13:54:30 lnxguill kernel: Unable to find swap-space signature
May 29 13:54:35 lnxguill kernel: eth0: no IPv6 routers present
May 29 13:54:46 lnxguill sshd[1256]: error: PAM: Authentication failure
for root from 9.57.138.243
May 29 13:54:49 lnxguill sshd[1256]: Accepted keyboard-interactive/pam
for root from 9.57.138.243 port 2063 ssh2
May 29 14:02:55 lnxguill sshd[1332]: Accepted keyboard-interactive/pam
for root from 9.57.138.243 port 2102 ssh2
```

The **head** command does the same thing, but displays only the first lines of a log file.

The **dmesg** command displays the messages logged by the kernel logger daemon.

Log files archiving

To avoid storing large amount of data, the log files are rotated on a regular basis. The rotation process is managed by the **logrotate** command, that is controlled by a *cronjob* launched daily.

For more information about *cronjob*, please refer to Chapter 5.4.1, "Job scheduling" on page 276.

Depending on the configuration options of logrotate, log files will be kept several days, weeks, compressed or not, when they reach a certain age or a certain size

As shown in Example 8-16, the older **messages** log file have been rotated 4 times, compressed and kept on disk. These were the defaults for SuSE Linux 9..

Example 8-16 Content of /var/log/ directory

```
mastervml2:/var/log # ls -al
total 3424
drwxr-xr-x   5 root root   4096 May 16 04:15 .
drwxr-xr-x  13 root root   4096 Jan 16  2007 ..
drwx------   2 root root   4096 Feb 12 15:10 YaST2
drwxr-x---   2 root root   4096 Feb 12 10:11 apache2
-rw-r-----   1 root root      0 Jan 16  2007 boot.log
-rw-r--r--   1 root root  13651 Apr 19 23:13 boot.msg
-rw-r--r--   1 root root  15623 Apr 19 23:12 boot.omsg
-rw-r--r--   1 root root    586 Jan 16  2007 convert_for_getconfig.log
-rw-------   1 root root  32096 Apr 10 00:15 faillog
-rw-r--r--   1 root tty  296888 May 27 21:27 lastlog
-rw-r--r--   1 root root    474 Dec  6 15:37 localmessages
-rw-r-----   1 root root   5334 Apr 30 17:20 mail
-rw-r-----   1 root root   1796 Apr 30 17:20 mail.err
-rw-r-----   1 root root   5334 Apr 30 17:20 mail.info
-rw-r-----   1 root root   3974 Apr 30 17:20 mail.warn
-rw-r-----   1 root root 214109 May 27 21:27 messages
-rw-r-----   1 root root 333920 Dec 11 04:15 messages-20071211.gz
-rw-r-----   1 root root 340177 Feb  2 04:15 messages-20080202.gz
-rw-r-----   1 root root 337732 Feb 21 04:15 messages-20080221.gz
-rw-r-----   1 root root 329943 May 16 04:15 messages-20080516.gz
[...]
```

In SuSE Linux 10, a new log file is created each time the current /var/log/messages file reaches 4Mb in size; log files are rotated 99 times before being removed. Logrotate and syslog configuration files can be found in /etc/logrotate.* and /etc/syslog.conf.

For more information about logrotate and syslog configuration, please refer to the corresponding manual pages.

Fine-tuning syslog and logrotate processes can be necessary to keep the right amount of log files according to your security / audit strategy.

Centralized remote logging

Last noteworthy feature of syslog is its full remote logging capability. syslogd is able to send messages to a remote host running syslogd and to receive messages from remote hosts.The remote host won't forward the message again, it will just log them locally.

Using this feature you're able to control all syslog messages on one host, if all other machines will log remotely to that. This tears down administration needs.

Please refer to syslogd manual page for more details.

8.3.2 Error analysis tools

This section discusses dumps and traces used as error analysis tools.

Dumps

If not enough relevant information can be found in the log files, in order to analyze the reasons one of the Linux guest running in a z/VM partition has a bad behavior, it can be useful to take a dump of the virtual machine for further analysis. Depending on the environment, several tools can be used:

- DASD dump tool, to generate a dump on 3390 or 3380 disks.
- SCSI dump tool, to generate a dump on FCP/SCSI devices
- Tape dump tool, to generate a dump on tape
- VMUDUMP, to create a dump of a guest virtual memory.

Refer to Table 8-2 for a list of the available tools.

Table 8-2 Available dump tools

Tool	Stand-alone tools			VMDUMP
	DASD	SCSI	Tape	
Environment	VM and LPAR	LPAR only	VM and LPAR	VM only
Speed	Fast	Fast	Slow	Medium
Medium	ECKD or FBA DASD	Linux file system on a SCSI disk	tape cartridges	VM reader
Compression possible	no	yes	yes	no
Disruptive	yes	yes	yes	no

Describing how to use all these tools is beyond the scope of this book. A detailed process for taking a dump is described in the “Using the dump tools” document. Its latest version can be downloaded from the Developerworks website at
`http://www.ibm.com/developerworks/linux/linux390/development_documentation.html#3`

Traces

Another useful way of identifying the failing step in the execution of a program is to trace the program. Linux provides a lot of different tools to trace the execution of a program. strace is one of these tools.

In the simplest case strace runs the specified command until it exits. It intercepts and records the system calls which are called by a process and the signals which are received by a process. The name of each system call, its arguments and its return value are printed on standard error or to the file specified with the -o option. Strace is a useful diagnostic, instructional, and debugging tool. System administrators, diagnosticians and trouble-shooters will find it invaluable for solving problems with programs for which the source is not readily available since they do not need to be recompiled in order to trace them. And programmers will find that, since system calls and signals are events that happen at the user/kernel interface, a close examination of this boundary is very useful for bug isolation, sanity checking and attempting to capture race conditions.

The output in Example 8-17 shows the beginning of the strace output when stracing the ls command. All system calls are details, with their parameters and the return codes sent. This can be used to identify which step of a program fails.

Example 8-17 stracing the ls command

```
lnxguill:/mnt/backup # strace ls
execve("/bin/ls", ["ls"], [/* 55 vars */]) = 0
brk(0)                                  = 0x80018000
mmap(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1,
0) = 0x2000001f000
uname({sys="Linux", node="lnxguill", ...}) = 0
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=33471, ...}) = 0
mmap(NULL, 33471, PROT_READ, MAP_PRIVATE, 3, 0) = 0x20000020000
close(3)                                = 0
open("/lib64/librt.so.1", O_RDONLY)     = 3
```

Note: Working closely with the development team of the application is necessary to analyze the output of the strace program.

This part gave a brief overview of how z/VM and Linux tools handle errors, and detailed some of the tools that can be used when trying to analyze why an application went wrong.

Sometimes, an abnormal program termination can lead to data corruption, for example if an application was writing some data to a disk when it crashed. In this case, the only option might be to restore data from an already existing backup, to come back to a state where the data is consistent. Backup and restore tools are described in the next part of the chapter.

8.4 Backup and restore

Backup refers to making copies of data so that these copies may be used to restore the original after a data loss event. These additional copies are typically called "backups." Backups are useful primarily for two purposes:

- the first is to restore a state following a disaster (called disaster recovery).
- the second is to restore small numbers of files after they have been accidentally deleted or corrupted.

Backups can be classified according to the way they are created:

- online backups - no interruption of operations.
- offline backups - disruptive, requires planning.

This chapter introduces some of the tools available for backing up and restoring z/VM and Linux data.

8.4.1 z/VM offline backups

Offline backups are disruptive operations, as they require the system to be stopped. This implies proper planning, as production has to be stopped.

Copying a z /VM system to disk.

Backing up a z/VM system to disk requires two different z/VM partitions, the one to backup and the one on which the commands will be issued. This also requires that the second partition has access to the disks of the z/VM partition to backup.

The process involves the following steps:

1. shutting down the z/VM system - system A - for backup
2. on the other system - system B - , varying system A disks online
3. from system B, copying system A disks using DDR - z/VM DASD Dump/Restore program - or Flashcopy (if enabled).
4. varying system A disks offline
5. reipling system A

This is for sure a disruptive process, all operations have to be stopped before the system is backed up.

> **Note:** Both z/VM and Linux can be saved to disk using this method.

Backing up a z/VM system to tape

As discussed in the previous chapter, this backup also requires two z/VM partitions, the second accessing the system disk of the first partition. The second partition also need access to a tape drive.

The process involves the following steps:

1. Attach the tape drive to z/VM system B
2. Rewind the tape
3. Vary online system A disks
4. Copy system A disks onto the tape, one after the other.
5. When done, detach the tape drive, and vary offline system A disks.

Note: Both Linux and z/VM disks can be saved to tape using this method.

Example 8-19 on page 381 is a script to perform a backup from disk to tape. Pre-requisites include a functional tape drive attached to z/VM as 181 device, and an input file DDR VOLUMES A (see Example 8-18) containing the list of DASDs to be backed up, each one on one line.

Example 8-18 DDR VOLUMES A input file format

```
8670
8671
8672
8673
8674
8675
8676
8677
8678
8679
```

Example 8-19 Example script to backup an environment to tape

```
/*DDR VOLUMES must contain input and output volumes */
Parse upper arg fn1 ft1 fm1
If fm1 = '' then fm1='A'
If fn1 = '' then do
   FN1='DDRTAPE'
   ft1='VOLUMES'
   fm1='A'
End
call traitfic
Traitement:
  do i = 1 to volumes.0 by 1
    'MAKEBUF'
    parse var volumes.i volin .
    SAY "DUMPING" VOLIN "ON TAPE DEVICE 181"
    QUEUE 'IN' volin '3390'
    QUEUE 'OUT 181 3590 (LEAVE LZCOMPACT'
    QUEUE 'DUMP ALL'
    QUEUE  'YES'
    QUEUE  ''
    'DDR'
    'DROPBUF'
  end
```

```
EXIT
traitfic:
  'STATE' fn1 ft1 fm1
  IF RC <> 0 THEN EXIT 99
    'pipe <',
       fn1 ft1 fm1,
       '| STEM  VOLUMES.'
return
```

Example 8-21 on page 382 is the script used to restore from tape to DASDs. Requirements are a tape drive attached to z/VM as 181 device, and a RESTAPES VOLUMES A (see Example 8-20) file with the list of target DASDs to restore to. It is possible to skip some DASDs in the restore process by specifying skip in the input file.

Example 8-20 Input file RESTAPE VOLUMES A

```
1800
1801
1802
1803
1804
skip
1806
1807
1808
1809
```

Example 8-21 Example script to restore an environment from tape to disk.

```
/* DDR volumes must be defined in file restape volumes a */
Parse upper arg fn1 ft1 fm1
If fm1 = '' then fm1='A'
If fn1 = '' then do
   FN1='RESTAPE'
   ft1='VOLUMES'
   fm1='A'
End
call traitfic
Traitement:
  do i = 1 to volumes.0 by 1
    if (volumes.i,1) = 'skip' then
   do
     SAY "Skipping 1 disk"
        TAPE FSF 1
```

```
   end
    else
      do
     parse var volumes.i volout .
        call restore
   end
  end
EXIT
restore:
  'MAKEBUF'
  SAY "RESTORING TAPE DEVICE 181 TO " VOLOUT
  QUEUE 'IN 181 3590 (LEAVE'
  QUEUE 'OUT' VOLOUT '3390'
  QUEUE 'RESTORE ALL'
  QUEUE 'YES'
  QUEUE 'YES'
  QUEUE ''
  'DDR'
  'DROPBUF'
  return
traitfic:
  'STATE' fn1 ft1 fm1
  IF RC <> 0 THEN EXIT 99
    'pipe <',
       fn1 ft1 fm1,
       '| STEM  VOLUMES.'
return
```

Backup of z/VM and Linux disks controlled from z/OS

It is possible to integrate z/VM and Linux into pre-existing z/OS backup and restore procedures.

Linux disks need to be prepared for use with the command **dasdfmt**. This command allows you to format a disk using a specific layout, Linux Disk Layout or Compatible Disk Layout.

Compatible disk layout is the default layout for dasdfmt. It means a special handling of the first two tracks of the volume by writing a VTOC on the disk. This enables other System z operating systems to access this device (e.g. for backup purposes).

This VTOC allows a Linux disk to be accessed from z/OS. This disk will be seen as a dataset called, for instance, LINUX.V0X0200.PART0001.NATIVE, and this dataset can be saved using standard z/OS DFSMSdss™ DUMP commands.

z/OS analogy: The following JCL is an example of a job to backup a Linux disk from z/OS using DFSMSdss dump:

```
//STEP1   EXEC PGM=ADRDSSU,REGION=0M
//IDIOT    DD  DISP=(NEW,CATLG),DSN=backup.dsn.........
//SYSPRINT DD  SYSOUT=*
DUMP INDY(lnxvol) OUTDD(IDIOT)
```

DFSMSdss dump can also be used to perform z/VM backup. In this case, it is necessary to do a physical dump, specifiying the tracks range to be copied.

z/OS analogy: The following JCL is an example of a job to backup a z/VM disk (3390-03) from z/OS using DFSMSdss dump:

```
//STEP1   EXEC PGM=ADRDSSU,REGION=0M
//IDIOT    DD  DISP=(NEW,CATLG),DSN=backup.dsn.........
//SYSPRINT DD  SYSOUT=*
   DUMP TRACKS(0,0,3338,14) -
   INDY(LX6RES) OUTDD(IDIOT) ADMINISTRATOR CPVOLUME
```

8.4.2 z/VM online backups

Online backups can also be called hot backups. They do not require the system to be shutdown before performing the backup.

Using SPXTAPE to backup spool files

It is possible to backup spool and system data files - that are printer, reader and punch files, as well as saved segments, NLS files, image librairies - to tape using the CP command SPXTAPE.

SPXTAPE command allows the operator to selectively backup to tape and restore to disk spool and system data files. All files can be dumped, or a filter applied to only save the matching files.

Copying z/VM CPOWNED minidisks

It is possible to perform a hot backup of a running z/VM system by copying z/VM system disks. This is a non disruptive backup, as the system continues to run while the disks are copied.

Each CPOWNED disk has a corresponding fullpack MDISK definition in MAINT user directory, as shown in Example 8-22.

Note: CPOWNED disks added after z/VM installation should also be added as FullPack MDISKs in MAINT USER DIRECTORY for consistency.

Example 8-22 CP owned disks, and corresponding MDISKs definitions.

```
Ready; T=0.01/0.01 15:35:36
q cpown
Slot  Vol-ID  Rdev  Type   Status
   1  LX6RES  1A20  Own    Online and attached
   2  LX6SPL  1A21  Own    Online and attached
   3  LX6PAG  1A22  Own    Online and attached
   4  LX6W01  1A23  Own    Online and attached
   5  LX6W02  1A24  Own    Online and attached

Ready; T=0.01/0.01 15:35:38
q v da
DASD 0122 3390 LX6SPL R/W       3339 CYL ON DASD  1A21 SUBCHANNEL =
000A
DASD 0123 3390 LX6RES R/W       3339 CYL ON DASD  1A20 SUBCHANNEL =
000B
DASD 0124 3390 LX6W01 R/W       3339 CYL ON DASD  1A23 SUBCHANNEL =
000C
DASD 0125 3390 LX6W02 R/W       3339 CYL ON DASD  1A24 SUBCHANNEL =
000D

Ready; T=0.01/0.01 13:35:12
```

These minidisks can be copied via DDR - or Flashcopy, if enabled - while the system is running, onto a new set of disks. Example 8-23 describes the set of inputs required to copy the first disk.

Example 8-23 Copying z/VM disks using DDR

```
ddr
z/VM DASD DUMP/RESTORE PROGRAM
ENTER:
IN 123 3390
ENTER:
OUT 1B34 3390
ENTER:
COPY ALL
HCPDDR711D VOLID READ IS LX6RES
DO YOU WISH TO CONTINUE?  RESPOND YES, NO OR REREAD:
YES
```

```
HCPDDR711D VOLID READ IS DK1B34
DO YOU WISH TO CONTINUE?  RESPOND YES, NO OR REREAD:
YES
COPYING   LX6RES
END OF COPY
ENTER:

END OF JOB
PRT FILE 0172 SENT FROM MAINT    PRT WAS 0172 RECS 0006 CPY  001 A
NOHOLD NOKEEP
Ready; T=0.09/1.13 13:41:07
```

All z/VM CPOWNED disks should be copied, with the exception of the paging disks that can simply be formatted and allocated as a paging device. This paging device must, however, have the same label than the existing paging device, otherwise the copied z/VM system will not IPL correctly.

Note: It is a good practice to change the system DASDs labels once you have copied your z/VM system. Refer to Chapter 4.3.3, “Relabeling the system volumes” on page 170 for more details.

This copy operation results in a brand new z/VM system ready to IPL. IPL will have to be FORCEd, as no Warmstart data was recorded. For more information about warmstart data, refer to Chapter 5.2.1, “z/VM IPL workflow” on page 230

Note: A couple of spool files - reader, printer or punch files - could have been lost during the copy process. This should not cause any problems restarting z/VM and the Linux virtual machines that were running in the environment.

This method cannot be applied to backups of running Linux machines. Backup of running Linux machines will lead to data inconsistency.

8.4.3 Linux backup tools

Linux distributions includes a lot of tools that can be used to backup and restore data. Some are very basic tools, such as tar or cpio for instance, some are more complex, such as IBM Tivoli Storage Manager.

This chapter introduces some of the tools provided by the Linux distributions to perform backups, but is by no means exhaustive.

Important: All the tools described in this part do not do any consistency checks or data integrity control.

Tar archiving utility

Tar is a standard Unix tool to create archives from a set of files or directories. This archive can then be compressed and saved onto another set of disk or a tape.

Example 8-24 Archiving files using tar

```
lnxguill:~ # tar cvf varlog_backup.tar /var/log
tar: Removing leading `/' from member names
/var/log/
/var/log/YaST2/
/var/log/YaST2/y2logRPM
/var/log/YaST2/y2log
/var/log/YaST2/y2logmkinitrd
/var/log/YaST2/y2log_bootloader
/var/log/YaST2/volume_info
/var/log/YaST2/disk_dasda
/var/log/YaST2/disk_dasdb
/var/log/YaST2/disk_dasdc
[...]
/var/log/scpm
/var/log/slpd.log
/var/log/boot.omsg
/var/log/dump/
lnxguill:~ # ls -al
total 5280
drwx------  5 root root    4096 May 29 14:36 .
drwxr-xr-x 22 root root    4096 May 29 13:54 ..
-rw-------  1 root root    2098 May 29 13:59 .bash_history
-rw-r--r--  1 root root    1332 Nov 23  2005 .exrc
drwx------  2 root root    4096 May 20 15:07 .gnupg
-rw-------  1 root root    1024 May 20 15:13 .rnd
-rw-------  1 root root    4072 May 28 10:12 .viminfo
drwxr-xr-x  2 root root    4096 May 20 15:14 .wapi
-rw-r--r--  1 root root   31732 May 20 15:14 autoinst.xml
drwxr-xr-x  2 root root    4096 Jun 16  2006 bin
-rw-r--r--  1 root root 5324800 May 29 14:36 varlog_backup.tar
```

The resulting file, varlog_backup.tar, can then be compressed and saved to another disk or to a tape.

Another use is to use tar to compress the amount of data to be sent to the backup server over ssh, as shown below:

Example 8-25 Backing up using tar and ssh

```
ceron:/var/log # tar cvf - /var/log | ssh 9.12.5.66 tar xvf -
tar: Removing leading `/' from member names
/var/log/
/var/log/YaST2/
/var/log/YaST2/y2logRPM
/var/log/YaST2/y2log
var/log/
var/log/YaST2/
var/log/YaST2/y2logRPM
var/log/YaST2/y2log
/var/log/YaST2/y2logmkinitrd
/var/log/YaST2/y2log_bootloader
[...]
var/log/zmd-messages.log.2008-05-24
var/log/zmd-messages.log.2008-05-25
var/log/zmd-messages.log.2008-05-26
var/log/zmd-messages.log.2008-05-27
var/log/zmd-messages.log.2008-05-28
```

The files in /var/log on ceron are compressed, sent via ssh to the backup server, and decompressed on the fly.

Disk dump - using dd command

The standard command dd can be used to perform a backup of a directory, a filesystem, a whole partition or a disk to a disk or to a file. The same command can be used to restore the dumped data.

Example 8-26 Backup using dd command

```
lnxguill:~ # dd if=/dev/dasdc1 of=/dev/dasdd1
3654528+0 records in
3654528+0 records out
1871118336 bytes (1.9 GB) copied, 181.17 seconds, 10.3 MB/s
```

Rsync

Rsync copies files either to or from a remote host, or locally on the current host (it does not support copying files between two remote hosts).

The first time it is called, rsync will do a full backup of a directory. All subsequent calls to rsync will only backup the modified files, hence reducing the time need for the backup.

Example 8-27 Backup with rsync command

```
lnxguill:~ # rsync -av /var/log/ /mnt
building file list ... done
./
boot.log
boot.msg
boot.omsg
faillog
lastlog
mail
mail.err
mail.info
mail.warn
messages
ntp
scpm
slpd.log
warn
wtmp
zmd-backend.log
zmd-messages.log
YaST2/
YaST2/disk_dasda
YaST2/disk_dasda-1
YaST2/disk_dasdb
YaST2/disk_dasdc
YaST2/disk_dasdc-1
YaST2/macro_inst_cont.ycp
YaST2/macro_inst_initial.ycp
YaST2/volume_info
YaST2/volume_info-1
YaST2/y2log
YaST2/y2log-1
YaST2/y2log.SuSEconfig
YaST2/y2logRPM
YaST2/y2log_bootloader
YaST2/y2logmkinitrd
YaST2/y2start.log
apparmor/
apparmor/reports-archived/
```

```
apparmor/reports-exported/
apparmor/reports/
audit/
audit/audit.log
cups/
dump/
krb5/
news/
news/news.crit
news/news.err
news/news.notice
smpppd/

sent 5282837 bytes  received 906 bytes  3522495.33 bytes/sec
total size is 5279388  speedup is 1.00
lnxguill:~ #

lnxguill:~ # rsync -av /var/log/ /mnt
building file list ... done
lastlog
messages
wtmp

sent 112424 bytes  received 86 bytes  225020.00 bytes/sec
total size is 5279901  speedup is 46.93
```

LVM2 snapshot

Snapshots are a feature of Linux Logical Volume Manager 2 that allow an administrator to create a new block device which presents an exact copy of a logical volume, frozen at some point in time.

Typically this would be used when some batch processing, a backup for instance, needs to be performed on the logical volume, but you don't want to halt a live system that is changing the data. When the backup of the snapshot device has been finished, the system administrator can just remove the device. This facility does require that the snapshot be made at a time when the data on the logical volume is in a consistent state

In Example 8-28 on page 391, a snapshot of the logical volume /dev/mapper/vgsystem-varloglv has been created. This logical volume holds the /var/log directory, and the snapshot /dev/mapper/vgsystem-varlogsnapshot presents the contents of this directory frozen at one point in time, in a consistent state, that allows to backup the data.

Example 8-28 Creating a LVM snapshot

```
lnxguill:~ # mount
/dev/dasda1 on / type ext3 (rw,acl,user_xattr)
proc on /proc type proc (rw)
sysfs on /sys type sysfs (rw)
debugfs on /sys/kernel/debug type debugfs (rw)
udev on /dev type tmpfs (rw)
devpts on /dev/pts type devpts (rw,mode=0620,gid=5)
/dev/dasdc1 on /usr type ext3 (rw,acl,user_xattr)
/dev/mapper/vgsystem-varloglv on /var/log type ext3 (rw,acl,user_xattr)
securityfs on /sys/kernel/security type securityfs (rw)
lnxguill:~ # lvcreate -s -L 3500MB -n varlogsnapshot
/dev/vgsystem/varloglv
  Logical volume "varlogsnapshot" created
lnxguill:~ # mount /dev/mapper/vgsystem-varlogsnapshot /mnt/backup/
lnxguill:~ # cd /mnt/backup/
lnxguill:/mnt/backup # ls -al
total 48
drwxr-xr-x  4 root root  4096 May 29 15:09 .
drwxr-xr-x  3 root root  4096 May 29 15:17 ..
-rw-r--r--  1 root root 11398 May 29 15:09 boot.msg
drwxr-xr-x 10 root root  4096 May 29 15:06 log
drwx------  2 root root 16384 May 29 15:05 lost+found
-rw-r-----  1 root root  1411 May 29 15:11 messages
-rw-r-----  1 root root   347 May 29 15:09 warn
```

For more details about Logical Volume Manager 2, refer to Chapter 5.3.9, “Managing DASD” on page 259

8.4.4 Other available tools

Backup and restore software are available from many vendors. Here are some examples:

- IBM Tivoli Storage Manager

  The IBM Tivoli Storage Manager's backup and recovery solution is a centralized, comprehensive solution that employs smart data moves and smart data store technology that makes backups and restores as fast, flexible and low-impact as possible. The IBM Tivoli suite of storage products supports more than a dozen OS platforms, a variety of network connectors and more than 500 offline storage devices.

For more information about IBM Tivoli Storage Manager, please refer to IBM Tivoli website http://www-306.ibm.com/software/tivoli/solutions/backup/

- CA VM:Backup

  CA VM:Backup provides an efficient and reliable means of backing up CMS and non-CMS data in z/VM, as well as Linux systems. End users can restore their individual files or minindisks while full volumes or entire systems can be restored by administrators. The CA VM:Backup Hidro feature adds stand alone backup and restore capability when no operating system is running on the processor.

  For more info about CA VM:Backup, please refer to this document on CA website: http://ca.com/files/ProductBriefs/vmbackup_pb2.pdf

- InnovationDP FDR/Upstream

  FDR/UPSTREAM offers you a powerful, reliable and high performance storage management product for centralized, automated, unattended backup/restore and archival for Open systems. Whether your data is stored on a LAN/NAS/SAN storage Upstream will provide automated operations to OS/390 or z/OS MVS mainframe server.

  FDR/UPSTREAM is the most simple, easy-to-use product to protect data. UPSTREAM lowers your TCO by using your existing z/OS and OS/390 systems and resources making it a perfect zLINUX storage management tool. For more information about FDR/UPSTREAM, please refer to the website: http://www.innovationdp.fdr.com/products/upstream/zlinuxups.cfm

Database vendors also provide their own database backup tools. For instance, Oracle® provides RMAN, Recovery Manager, to handle backup and restore of their database, whatever the platform it is running on.

9

Applying system maintenance

Servicing z/VM does not differ much from servicing a z/OS system. The tasks involved are to a great extent identical for both platform. They even have many of the terms in common. From a z/OS perspective, however, there are some differences beside the fact that z/VM in some areas has brought virtualization to a higher degree than z/OS.

The first thing to consier is that service is performed from a virtual machine, normally in a much more interactive manner than what's considered common practice on z/OS. At the same time, the tools provided on z/VM are often more automated during the entire process.

The aspect that might differ the most from a z/OS perspective, is the fact that z/VM does not have the concept of manageable data sets (or even disk volumes from this perspective) to the same extent as z/OS has. z/VM has some architectural differences from z/OS when it comes to the relationship between the system residence volume(s) and files containing parameterization (for instance **SYSTEM CONFIG** and **USER DIRECT** files). By design, the callable services or executables and parameterization are more bundled or tied to each other on z/VM, compared to z/OS. This differs from z/OS, where you *could* keep the SYS1.PARMLIB concatenation and RACF (or similar) user definintions isolated from the system residence volumes. Placing and referencing of data sets could also be managed by symbols in z/OS (both parameterization and catalog) to support distribution of system residence at volume level.

The **BUILD** process is, in this respect, the one being most different from a z/OS perspective. This process could actually be seen as the processes one would perform to distribute a serviced z/OS system. Also the fact that you actually service the system from within itself without having an obvious method or path to distribute for testing, is a big difference. In z/OS, you would typically install service to dedicated disks and then distribute from those.

There is also a difference in how z/VM uses it’s PARMLIB counterpart, the SYSTEM CONFIG file as opposed to how z/OS is using the **PARMLIB**. SYSTEM CONFIG is only read during the IPL process, whereas the **PARMLIB** is also read for certain dynamic updates. Based on z/VM’s possibilities to do dynamic changes without IPLs, there should be a (even more so compared to z/OS) need to establish suitable processes to keep or track changes provided they are intended to be permanent.

This chapter discusses the processes involved in maintaining a z/VM system in comparison with performing the similar processes on z/OS. It is limited to the technical aspect of these processes, and does not cover any administrative processes or routines.

Objectives:

Upon completion of this chapter, you should be familiar with:

- Maintenance methods provided by IBM for a z/VM system
- How the maintenance is brought into production
- Similarities and differences between z/VM and z/OS maintenance

9.1 z/VM maintenance methods

Maintenance of a z/VM system always implies logging onto a virtual machine that has the neccessary privilege classes to perform the maintenance. The actual install activities are performed from within that virtual machine. By default, the designated virtual machine to perform system maintenance would be **MAINT**, but z/VM sites *could* establish other virtual machines dedicated for such activities.

The maintenance is normally performed using the **Virtual Machine Serviceability Enhancements Staged/Extended** (**VMSES/E**) product, which should be seen as the z/VM equivalent of **System Modification Product/Extended (SMP/E)** in z/OS (providing the same set of services). This product includes a set of tools in the form of EXECs that performs the tasks necessary to maintain the z/VM system. Due to z/VM nature, the process of maintaining a z/VM system is a more interactive task, compared to z/OS. z/OS personnel would probably prepare and run a batch jobs to perform these tasks.

9.2 VMSES/E terminology

This section explains some of the terms used for z/VM maintenance.

9.2.1 Deliveries

General service or maintenance to a z/VM system is usually divided into the following classifications of delivery:

- Recommended Service Upgrade (RSU)
- Corrective Service (COR)
- Expanded Service Option (ESO): A defined collection of service in COR format. Could include service within starting and ending service levels, or could be more specific or customized regarding types of service or customized to your profile.

A downloaded file containing service is often refered to as an **envelope**.

9.2.2 Element terms

The terms ***authorized program analysis report*** (APAR) and ***program tempory fixes*** (**PTF**) are used used within VMSES/E having the same meaning or definition as in SMP/E.

The term ***small programming enhancement*** (SPE) is used for service for installing a new release or version of a product. This corresponds to the SMP/E term **FUNCTION**..

Any service not supplied by IBM on a **COR** or RSU service tape (or other media) are defined using the term ***local service*** or ***local modifications***. This would by definition include:

- Service (for instance PTFs) you have been forced to install due to experienced errors or similar between COR or RSU cycles
- Any modifications caused by third party software installations
- Any local modifications you have installed to tailor your system.

> **z/OS analogy:** A comparison to SMP/E for z/OS would be that the latter two could be similar to **USERMOD**s. The PTFs installed ‘in between’ CORs and RSUs would have no special definition attached to them.

The ***VMSES/E Introduction and Reference*** manual strongly advise the use of ***local tracking number*** for such service in order to manage them properly. This would be similar to the use of **SOURCEID** in SMP/E (could be assigned during RECEIVE and used for further logical processing of such service).

9.2.3 BUILD process

Whenever there is a need to update the z/VM Control Program (**CP,** also named NUCLEUS) or **shared segments**, this is performed by a **BUILD** process. This will require an IPL if concerning the CP. A z/OS analogy is service applied to SYS1.NUCLEUS.

If it merely concerns the shared segments, it would require a restart of the affected virtual machines. A z/OS analogy would be a TSO/ISPF user logging off and on after a LINKLIST rebuild affecting one of the user’s ISPF applications.

9.3 Comparing VMSES/E and SMP/E

VMSES/E is normally invoked by executing **VMFINS** using parameters for the different types of tasks to perform. However, other executables are provided to perform more coumpound activities (chained, logically controlled execution of more than one VMSES/E exec). These executables perform much like a batch job having multiple steps and conditional execution.

For detailed information on how to use VMSES/E, see z/VM VMSES/E Introduction and reference, GC24-6130

VMSES/E like SMP/E has its software inventory. This is by default stored on minidisks owned by MAINT.

9.3.1 Installing COR service

Installing service from a COR tape (or other media) is very similar to performing a SMP/E **RECEIVE/APPLY** process before putting to production. One should:

1. Read the MEMO-to-Users
2. Use the VMFREC command to receive the service
3. Review any messages from VMFREC and correct them
4. Verify the installation environment
5. Run VMFAPPLY to apply the service
6. Review any messages from VMFAPPLY and correct them
7. Perform any tasks required to (re)build the component
8. Review any messages and correct errors
9. Test the serviced components and correct any errors
10. Build any saved segments
11. Merge the intermediate apply disk with the tested service to production

All these tasks, with the exeption of the review and correct items, could be performed using the coumpound execs **SERVICE** and **PUT2PROD**. See also Figure 9-1 on page 398

Items 1-3 corresponds to a **SMP/E RECEIVE** process, 4-6 to the **APPLY** process and 7-11 would include any distribution and/or other activities you would normally perform outside of SMP/E, depending on your procedures. For some z/OS sites, SMP/E **ACCEPT** could be part of such procedures.

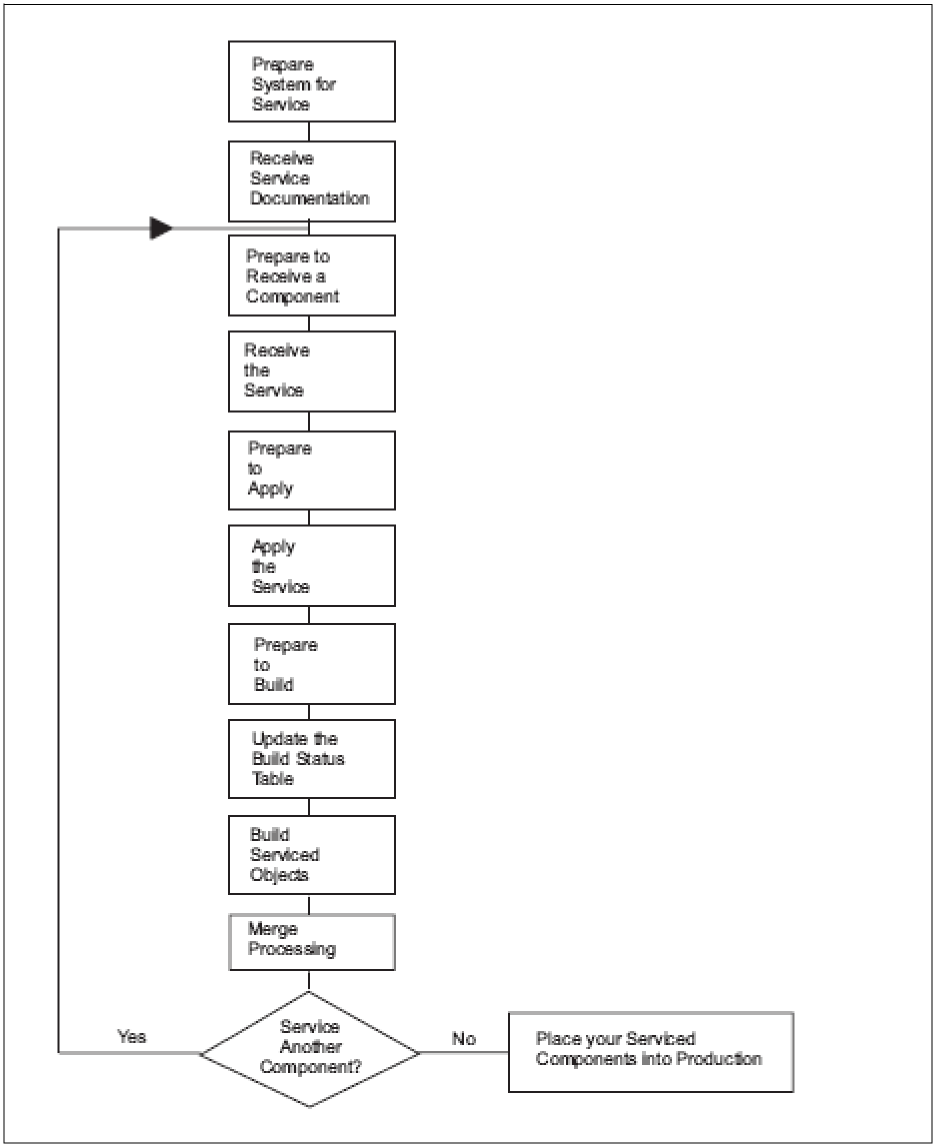

Figure 9-1 Cor installation flow, taken from VMSES/E Introduction and Reference manual

9.3.2 Product Service Upgrade (PSU)

This type of maintenance is used for servicing or maintaining your z/VM portfolio to a higher service level (not for upgrading). The PSU procedure use the

Recommended Service Upgrade (RSU) media to accomplish this. The media has a fixed logical structure per product, shown in Figure 9-2 (taken from z/VM:VMSES/E Introduction and Reference).

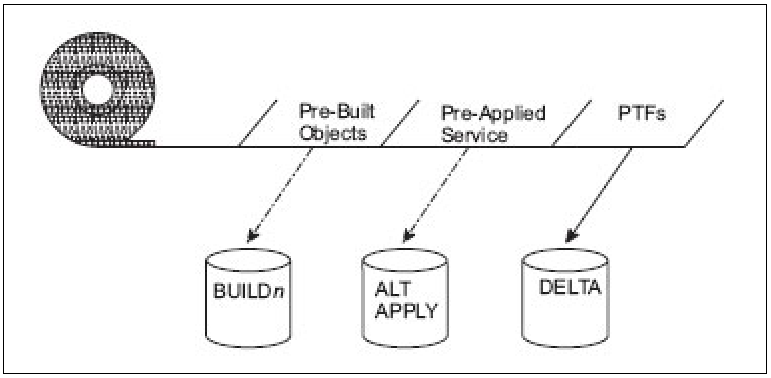

Figure 9-2 Logical file structure of RSU media

The process is performed in the following order:

1. Receive and view documentation for the product
2. Prepare to receive the product using exec VMFPSU to help you plan
3. Receive from RSU
4. Process additional service for the product not included in RSU, including any l**ocal modifications**
5. Rebuild the product
6. Repeat 1-5 for every product until all are completed

> **Note:** Item 4) would be similar to reapplying usermods that were **REGRESSED** by service applied in **SMP/E**

Figure 9-3 on page 400 illustrates the RSU installation tasks

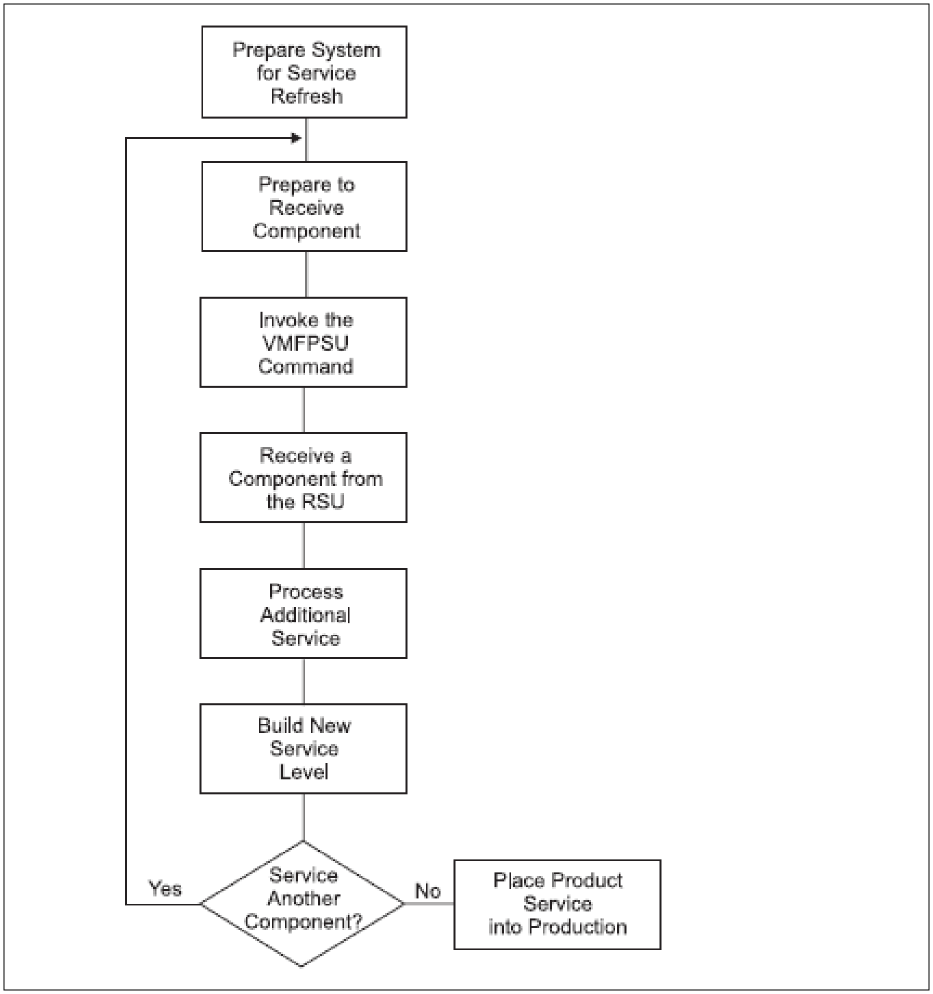

Figure 9-3 PSU installation work flow

9.4 Convenient maintenance practices for z/VM

A convienient practice for servicing your z/VM system would be to bring up (or clone) a secondary level z/VM for this purpose. From within this clone or copy, you could perform any service actvities on that system. That would provide a suitable environment for testing prior to performing identical servicing of your first level system. Many z/VM sites do their servicing in this way. This maintenance environment could be accomplished by installing a new z/VM from a virtual machine defined to your first level z/VM. You could also clone your initial installation, provided a backup copy exists and the disks containing the original installation has been relabelled (to avoid potential duplicate volser situations).

For details on how to install a secondary z/VM, refer to manual z/VM: Guide for Automated Insallation and Service, GC24-6099

The scenario of having duplicate labelled disk volumes is further discussed in the following publications:

z/VM and Linux on IBM System z: The Virtualization Cookbook for SLES9, SG24-6695

IBM z/VM and Linux on IBM System z: Virtualization Cookbook for Red Hat Enterprise Linux 4, SG24-7272

9.5 Case study

During the writing of this book, a second level z/VM was brought up under the primary one. Service was downloaded from the web, and then FTPed to MAINT's 500 virtual disk on the system. These downloaded files were DETERSEd to MAINTs 500 virtual disk, this time having a filetype of SERVLINK. One should also verify that these files have a record length of 4005.

> **Note:** All these activities were performed from the MAINT virtual machine

Example 9-1 on page 401 shows a terminal display after invoking the DETERSE exec for each of the downloaded files.

Example 9-1 DETERSE of the downloaded files

```
Ready; T=0.01/0.01 09:28:23
deterse S7820380 SHIPRSU1 H1 530RSU03 SERVLINK H
Ready; T=2.04/2.15 09:29:42
deterse S7820381 SHIPTFSS H1 530SERVI SERVLINK H
Ready; T=0.16/0.17 09:30:12
```

After having prepared the files, the following commands were issued from the virtual machine for the secondary z/VM (named VMGUEST) after detaching 500 from the primary's MAINT machine:

1. **VARY ONLINE 500**
2. **ATT 500**

3. **ACC 500 h**
4. **FILEL * * h** should display the DETERSEd files

Then we issued the SERVICE command to install the RSU:

SERVICE ALL 530RSU03

Example 9-2 on page 402 and Example 9-3 on page 402 show the terminal output of the the invoked SERVICE exec

Example 9-2 Output from SERVICE exec

```
VMFINS2603I Processing product :PPF SERVP2P REXX :PRODID 5VMREX30%REXX
VMFREQ2805I Product :PPF SERVP2P REXX :PRODID 5VMREX30%REXX has passed
requisite
           checking
VMFINT2603I Installing product :PPF SERVP2P REXX :PRODID 5VMREX30%REXX
VMFREC2760I VMFREC processing started
VMFREC1852I Volume 1 of 1 of INS ENVELOPE 5303
VMFREC1851I (1 of 4) VMFRCAXL processing AXLIST
VMFRCX2159I Loading 4 part(s) to DELTA 3D2 (K)
VMFREC1851I (2 of 4) VMFRCPTF processing PARTLST
VMFRCP2159I Loading 1 part(s) to DELTA 3D2 (K)
VMFREC1851I (3 of 4) VMFRCCOM processing DELTA
VMFRCC2159I Loading 3 part(s) to DELTA 3D2 (K)
VMFREC1851I (4 of 4) VMFRCALL processing APPLY
VMFRCA2159I Loading part(s) to APPLY 3A6 (G)
VMFRCA2159I Loaded 5 part(s) to APPLY 3A6 (G)
VMFREC2189I Updating Requisite table 5VMREX30 SRVREQT, Description
table
            5VMREX30 SRVDESCT and Receive Status table 5VMREX30 SRVRECS
with 1
          new PTFs from INS 5303




                                                              VM READ
VMGUEST
```

Example 9-3 More output from SERVICE exec

```
VMFUTL2205I BUILD9    R      R/W  402   MNT402
VMFUTL2205I BASE2     T      R/W  6B2   MNT6B2
```

```
VMFUTL2205I BASE4    U     R/W  3B2   MNT3B2
VMFUTL2205I --------  A     R/W  191   MNT191
VMFUTL2205I --------  B     R/W  5E5   MNT5E5
VMFUTL2205I --------  C     R/W  2CC   MNT2CC
VMFUTL2205I --------  D     R/W  51D   MNT51D
VMFUTL2205I --------  S     R/O  190   MNT190
VMFUTL2205I --------  Y/S   R/O  19E   MNT19E
VMFSET2760I VMFSETUP processing completed successfully
AUTO LOGON  ***      BLDNUC   USERS = 14
BLDNUC  : CONNECT= 00:00:01 VIRTCPU= 000:00.03 TOTCPU= 000:00.06
BLDNUC  : LOGOFF AT 09:50:19 EDT FRIDAY 05/23/08 BY MAINT
USER DSC   LOGOFF AS  BLDNUC   USERS = 13    FORCED BY MAINT
VMFSET2760I VMFSETUP processing started for DETACH GCS
VMFSET2760I VMFSETUP processing completed successfully
VMFSRV2760I SERVICE processing completed successfully for GCS BUILD
VMFSUT2760I VMFSUFTB processing started
VMFSUT2760I VMFSUFTB processing completed successfully
VMFSRV2760I SERVICE processing completed successfully
Ready; T=28.63/30.61 09:50:31
CMS

                                                     RUNNING   VMGUEST
```

9.5.1 Invoking PUT2PROD

After having performed the previous steps, we run the exec **PUT2PROD**. We had trouble with one of the PTFs during PUT2PROD processing. It turned out that this particular problem was documented as a known error, also containing the necessary actions to circumvent the problem. After we had performed these documented actions, we succeeded running the PUT2PROD exec.

Upon completion, we issued

– SHUTDOWN REIPL

from MAINT. Issuing command

– **Q CPLEVEL**

displayed that the CP level is now at a higher RSU level.

9.5.2 Installing additional PTFs

We then applied the additional PTFs (shipped since the RSU was released). They should have been installed using SERVICE prior to PUT2PROD to avoid the first problem we ran into. The INSTALL of these PTFs did not complete successfully. It turned out to be that the 500 virtual device should be in read/write mode (R/W), probably due to some temporary build process using that disk during the SERVICE process. After correcting this, the SERVICE completed succesfully and so did PUT2PROD.

Important: Please note, apply PTFs ***prior*** to using PUT2PROD to avoid errors in PUT2PROD.

Ensure that the 500 virtual device is in R/W mode prior to installing PTFs

9.6 Putting your serviced z/VM to production

After maintaining your z/VM with service, an IPL is required (as it could be service to CP). IPL the system whenever it is convenient, depending on your service hours and/or other conditions. Like any other operating system, all users (in this case all virtual machines) will be shut down. z/VM has the ability to perform a **SHUTDOWN REIPL,** provided it is issued from a virtual machine having the proper privilege class to do so[1]. Virtual machines required to run at all times, could be **AUTOLOG**'ed during IPL.

[1] Does not have to be performed from an HMC console as opposed to a z/OS IPL

A

z/OS to z/VM and Linux command tables

This appendix provides a cross reference betwen some z/OS commands and concepts and their z/VM and Linux counterparts. While not all commands and concepts are listed here, we feel this is a good start for the z/OS systems programmer in becoming more familiar with z/VM and Linux on System z.

For our examples here, we refer to SLES10 commands when we had to be specific. We recommend checking your Linux distribution to verify the commands remain the same.

Table A-1 Commands for file editing

z/OS-*ISPF Edit*	z/VM CMS XEDIT	Linux vi	Description
r	"		repeat/copy a line
d	d	dd	delete a line
cc...cc line command	cc ...cc	`:i,coj`	Block copy
mm...mm line command	*mm...mm*	`:i,jmk`	block move
dd...dd	*dd...dd*	`:i,jmk`	block delete

find <xxxxx>	*/<xxxxx>*	/<xxxxx>	find keyword (below)
find <xxxxx> prev	*-/<xxxxx>*	?<xxxxx>	find keyword (above)
ex xxx			exclude lines containing a character string (xxx)
del all excluded			delete the lines that you excluded using the command “ex xxx”
))n line command			indent ‘n’ spaces to the right
((n line command			move ‘n’ spaces to the left

Table A-2 General commands

z/OS command	**z/VM command**	**Linux command**	**Description**
d u,,,b120,1	*q b120*	*ls -la /dev/<device>*	display a single device named b120 (replace b120 with your device)
d u,,,b120,8	*q b120-b127*	*ls -la /dev/<sd*>*	display string of devices
v b120,online	*vary on b120*	*mount /dev/<device> <mnt_point>*	vary on a device
v b120,offline	vary off b120	*umount mnt_point*	vary off a device
	q paths b120		display paths to a device
	q chpid 18		display devices on a chpid
D M=CPU	q proc	*cat /proc/cpuinfo*	display number of processors

Table A-3 FTP and TCP/IP commands

z/OS command	**z/VM command**	**Linux command**	**Description**
FTP hostname {port}	FTP hostname {port} *Enter FTP ? for list of options*	ftp hostname <-P port>	Connect to remote host to get/put files. Defaults to port 21

HOMETEST	HOMETEST	*ping <gateway ip address>*	Validate TCP/IP configuration
NETSTAT *option {TCP procname}*	NETSTAT *option*	netstat	Display network status of local host. Use ? for list of options

Table A-4 General concepts

z/OS	**z/VM**	**Linux**
IPL(initial program load)	IPL(initial program load)	boot
Nucleus	Control Program (CP)	kernel
HELP command	HELP command	man command
Data set SYS1.UADS	user directory	user registry
TSO or ISPF edit	XEDIT	vi
TSO or ISPF	Conversational Monitor System (CMS)	shell

B

Planning worksheet

Fill in this worksheet in order to store the values you plan to use for your z/VM and Linux installation on System z.

Table B-1 Reader's resources worksheet

Name	Value	Comment
LPAR name		
CPC name		
z/VM system name		
TCP/IP host name		
TCP/IP domain name		
TCP/IP gateway		
DNS server 1		
DNS server 2/3 (optional)		
OSA device name		
OSA starting device number		
TCP/IP address		

Name	Value	Comment
Subnet mask		
OSA device type		
Network type		
Port name (optional)		
Router type		
Primary OSA device number for VSWITCH		
Secondary OSA device number for VSWITCH		
DASD addresses for z/VM		
DASD addresses for Linux		
DASD addresses for paging		

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get Redbooks” on page 412. Note that some of the documents referenced here may be available in softcopy only.

- *Linux on IBM System z: Performance Measurement and Tuning,*SG24-6906-01
- *Introduction to the New Mainframe: z/VM Basics*, SG24-7316
- *Security on z/VM*, SG24-7471

Other publications

These publications are also relevant as further information sources:

- *z/VM: VMSES/E Introduction and Reference,* GC24-6130
- *z/VM: Guide for Automated Installation and Service,* GC24-6099
- *z/VM: CMS User’s Guide, SC24-6079*
- *z/VM: CP Planning and Administration, SC24-6083*
- z/OS: MVS Initialization and Tuning Reference, SA22-7592
- *Melinda Varian, VM and the VM community, past present, and future,* SHARE 89 Sessions 9059–61, 1997
- *z/VM V5R3.0 Diagnosis Guide, GC24-6092*
- *z/VM V5R3.0 CP Messages and Codes, GC24-6119*
- *Virtual Machine Operation, SC24-6128*
- *z/VM V5R3.0 System Operation, SC24-6121*
- *EREP V3R5 User’s Guide, GC35-0151*

Online resources

These Web sites are also relevant as further information sources:

- z/VM library

  `http://www.vm.ibm.com/library/`

How to get Redbooks

You can search for, view, or download Redbooks, Redpapers, Technotes, draft publications and Additional materials, as well as order hardcopy Redbooks, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5" spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

IBM Redbooks

z/VM and Linux Operations for z/OS Systems Programmers

(1.5” spine)
1.5”<-> 1.998”
789 <->1051 pages

IBM Redbooks

z/VM and Linux Operations for z/OS Systems Programmers

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

IBM Redbooks

z/VM and Linux Operations for z/OS Systems Programmers

(0.5” spine)
0.475”<->0.875”
250 <-> 459 pages

IBM Redbooks

z/VM and Linux Operations for z/OS Systems Programmers

(0.2”spine)
0.17”<->0.473”
90<->249 pages

(0.1”spine)
0.1”<->0.169”
53<->89 pages

To determine the spine width of a book, you divide the paper PPI into the number of pages in the book. An example is a 250 page book using Plainfield opaque 50# smooth which has a PPI of 526. Divided 250 by 526 which equals a spine width of .4752". In this case, you would use the .5” spine. Now select the Spine width for the book and hide the others: **Special>Conditional Text>Show/Hide>SpineSize(-->Hide:)>Set** . Move the changed Conditional text settings to all files in your book by opening the book file with the spine.fm still open and **File>Import>Formats** the Conditional Text Settings (ONLY!) to the book files.

IBM

Redbooks

z/VM and Linux Operations for z/OS Systems Programmers

(2.5” spine)
2.5”<->nnn.n”
1315<-> nnnn pages

IBM

Redbooks

z/VM and Linux Operations for z/OS Systems Programmers

(2.0” spine)
2.0” <-> 2.498”
1052 <-> 1314 pages

Draft Document for Review July 30, 2008 9:05 am

z/VM and Linux Operations for z/OS Systems Programmers

IBM®

Redbooks®

An introduction to z/VM and Linux to z/OS programmers

Migrate z/OS to a z/VM guest system

Install and manage Linux on System z

This IBM Redbooks publication discusses z/VM and Linux operations from a z/OS programmer or system programmer's perspective. While there are other books written about many of these topics, you will find that this book gives enough information about each topic to describe z/VM and Linux on System z operations to somebody new to both.

This book is intended for z/OS programmers and system programmers who are transitioning into the z/VM and Linux on System z environments and need a translation guide to assist them.

We base this book on our experiences using System z10™ Enterprise Edition, z/VM version 5.3 RSU 0701 and Novell® Suse Linux Enterprise Server (SLES) 10 on System z.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7603-00 ISBN